## Letter of Transmittal

*3 May 2024*

*Remote Rehabilitation Team*

*Xian Gao, Molly Meadows, Noah Rieth*

*The University of Idaho Computer Science Department,*

*Enclosed is the final report of our research and development of the “Using Deep Learning to Provide Feedback for Remote Physical Rehabilitation” project sponsored by your department, Dr. Aleksandar Vakanski, and Dr. Min Xian.*

*In this report we detail our validation and results when testing three machine learning approaches we developed through this academic school year. We concluded that the recurrent neural network model performed the best for this task.*

*For any questions regarding the details in this report or further questions about this project, please contact Molly Meadows at* [*mead9103@vandals.uidaho.edu*](mailto:mead9103@vandals.uidaho.edu) *or at 208-741-3016.*

*Thank you,*

*Remote Rehabilitation*

## Cover Page

*Using Deep Learning to Provide Feedback for Remote Physical Rehabilitation*

*Remote Rehabilitation*

*Xian Gao, Molly Meadows, Noah Rieth*

*Instructor: Bruce Bolden*

*Sponsored by The University of Idaho Computer Science Department*

*for Dr. Aleksandar Vakanski, Dr. Min Xian*

*Contact Information: Molly Meadows* [*mead9103@vandals.uidaho.edu*](mailto:mead9103@vandals.uidaho.edu) *208-741-3016*

## Table of Contents

[Letter of Transmittal 1](#_Toc1150930662)

[Cover Page 1](#_Toc939445385)

[Table of Contents 2](#_Toc1353435299)

[Executive Summary 3](#_Toc2129089656)

[Background 4](#_Toc32711305)

[Problem Definition 5](#_Toc661852401)

[Project Plan 6](#_Toc1129892826)

[Concepts Considered 8](#_Toc751798230)

[Concept Selection 12](#_Toc758856504)

[System Architecture – 2+ pages 13](#_Toc623843898)

[Design Evaluation 20](#_Toc1480104063)

[Future Work 25](#_Toc694569399)

[Citations 32](#_Toc745372056)

## Executive Summary

Doctors recommend that patients complete physical therapy exercises at home, to regain mobility and strength following an injury. But without real-time feedback to correct flawed movements, patients may not recover in the expected time. To remedy this issue, we developed several machine learning models to provide accuracy scores for videos taken of physical rehabilitation exercises.

Two of the deep learning models were developed from a Spatio-Temporal model previously proposed and published by our clients. One development of this model included changing the input from 3D angular data captured by Vicon to 3D positional data captured by both Vicon and Kinect. We found that the Vicon positional data performed better than the Kinect positional data. The other development of this model changed the input to 2D positional data collected from a third-party human keypoint detection model, OpenPose, that extracts skeletal joint positions from any video file.

The third model consisted of a basic recurrent neural network system, to classify an exercise, and then predict an accuracy score using joint positions collected from OpenPose.

The recurrent neural network model had a higher accuracy when predicting if a video was an incorrect version of an exercise.

## Background

Patients experience frustration with physical rehabilitation recovery time due to many factors. Sometimes, this is because the patient unknowingly performs the physical therapy exercises incorrectly or ineffectively. Occasionally, patients are discouraged from completing their at-home exercises due to lack of clear instruction or feedback on how to perform them. Additionally, traditional physical therapy sessions can be unaffordable or inaccessible to many people. To mitigate this real-world problem, the University of Idaho Computer Science Department sponsored our team to research and develop ways to improve the practicality and cost effectiveness of rehabilitation practices through deep learning. Our team explored two machine learning models to provide feedback for physical rehabilitation exercises to patients remotely. A patient would upload a video of them completing one repetition of their physical therapy exercise taken from their smartphone from the comfort of their own home. Our models analyze the video using OpenPose, and then use machine learning to predict an accuracy score for that exercise. This preliminary work will lead to benefits for medical professionals in the form of freeing up time that is currently spent monitoring patients and will lead to benefits for patients by reducing rehabilitation costs.

## Problem Definition

The goal of our project is to develop a deep learning system to evaluate the performance of physical rehabilitation exercises. The input into the system would be a video of a single repetition of an exercise taken and uploaded by a smart phone. Our system would analyze the video using another convolutional neural network named OpenPose to extract 25 skeletal joint positions per frame of the video. We then would apply data preprocessing techniques to reduce noise and ensure that every input contains the same amount of data.

Project Deliverables:

|  |  |
| --- | --- |
| ***General Requirement:*** | ***Testing metric:*** |
| Analyze a smart phone video | Use OpenPose to locate joint positions from an MP4 or AVI video. |
| Provide an accuracy score | Scale of 0 – 1 (1 being a perfect repetition, 0 being not a repetition of that exercise) |
| Returns accuracy score in a reasonable amount of time < 20 minutes | Time from upload to accuracy score output in seconds |
| Extract movement data from joint positions | Store information extracted from OpenPose in machine readable format (CSV) |
| Model must be trained on at least one exercise | Train model on Deep Squat exercise |

Additional tasks include:

* Train the machine learning models on more than one exercise:
  + Inline Lunge
  + Side Lunge
* Adjust the current Spatio-Temporal to use the Vicon and Kinect positional data instead of angular data
* Compare the accuracy of quality scores from skeletal data obtained through different sources: OpenPose, Vicon, Kinect
* Train a model to recognize which type of exercise is being performed

## Project Plan

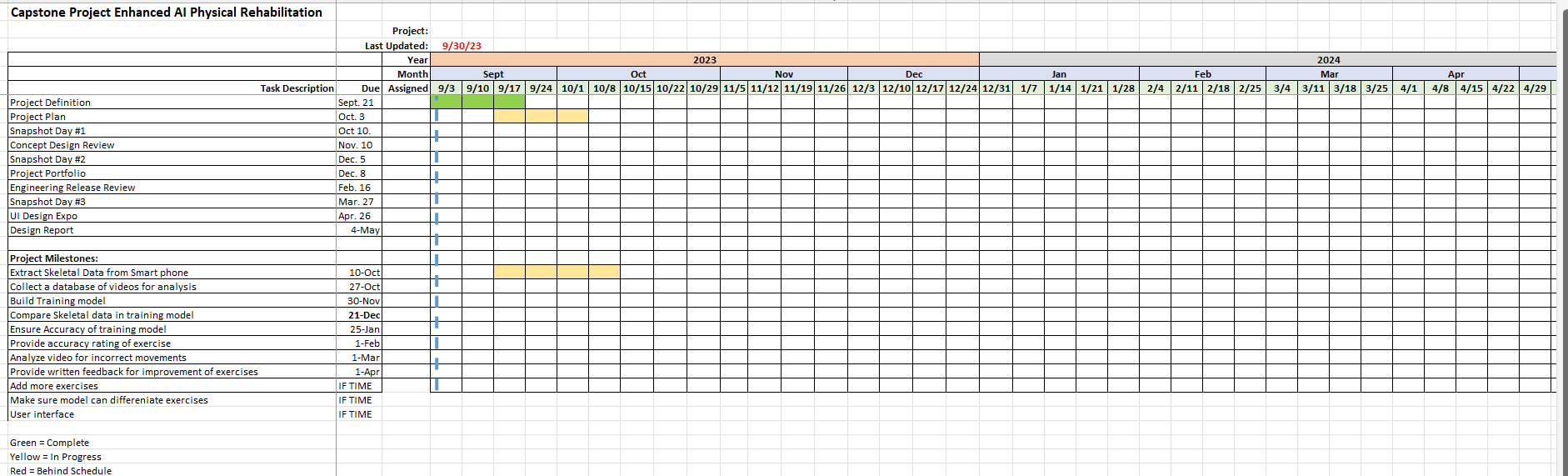
Our team consists of three undergraduate Computer Science students at the University of Idaho: Xian Gao, Molly Meadows, and Noah Rieth.

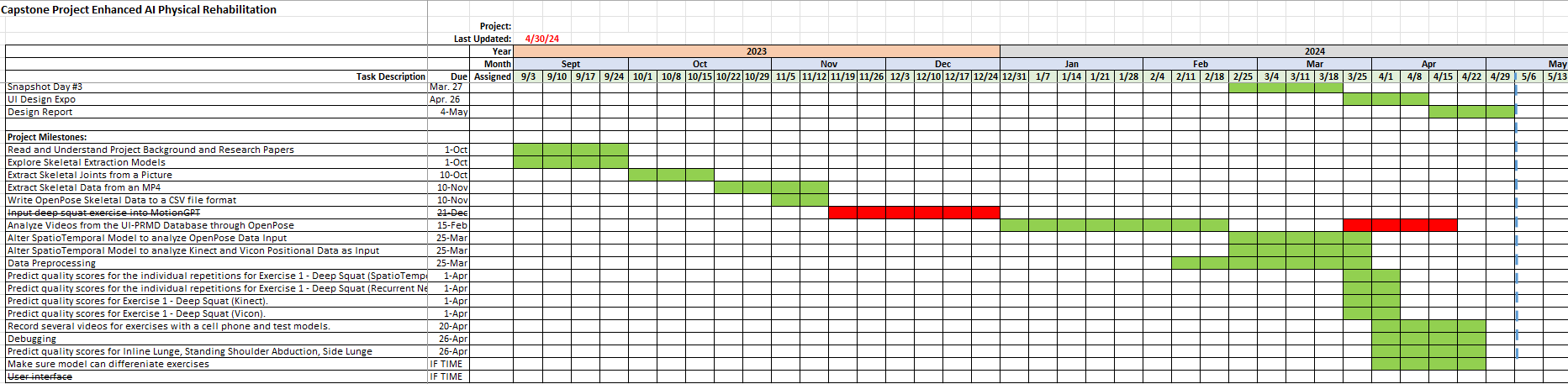
Molly was designated as the team leader of the group, who organized team meetings and created agendas for two-week periods. Molly was also responsible for organizing the OneDrive and GitHub repository where all the project information is stored. Molly explored and amended the Spatio-Temporal model that was originally proposed and published by our clients. She developed this model to produce accuracy scores for the deep squat, inline lunge, and side lunge exercises, using the OpenPose data output of these exercises as input to the models.

Noah was the primary contact between our client, Dr. Alexsandar Vakanski, and our team. He organized meeting times with our client and was the communicator for any questions or concerns that we had. Noah created a recurrent neural network to classify each input video to the correct exercise out of 10 different exercises. After classification, another recurrent neural network model was trained and used to produce an accuracy score for the given exercise.

Xian was responsible for developing the Spatio-Temporal model to use Kinect and Vicon positional data to produce accuracy scores for the exercises.

For tracking our progress and maintaining a schedule, we created a Gantt chart. One half of the document was to track due dates and progress for class deliverables and the other to plan and track specific project milestones. Refer to Figure 1 for details of our initial plan. In January, the scope of our project was redirected. Our original concept using MotionGPT (see in concepts considered) was not feasible to complete in our given timeline and project requirements. The schedule for the class deliverables stayed the same, but the milestones had to be adjusted. Refer to Figure 2 for details of our final version of the schedule.

Figure 1: Schedule in a Gantt Chart format at the beginning of the project. [Gantt Chart](https://vandalsuidaho-my.sharepoint.com/:x:/r/personal/mead9103_vandals_uidaho_edu/_layouts/15/Doc.aspx?sourcedoc=%7BA933D7DB-CC75-4898-B3FC-88EA1350B697%7D&file=GANTT_Schedule.xlsx&action=default&mobileredirect=true&wdOrigin=METAOS.FILEBROWSER.FILES-HOME)

Figure 2: Schedule in a Gantt Chart format at the end of our project. [Gantt Chart](https://vandalsuidaho-my.sharepoint.com/:x:/r/personal/mead9103_vandals_uidaho_edu/_layouts/15/Doc.aspx?sourcedoc=%7B095660DF-A51E-473D-B42A-5F1C0A75FAD4%7D&file=GANTT_Schedule%20(version%202).xlsx&action=default&mobileredirect=true&wdOrigin=METAOS.FILEBROWSER.FILES-HOME)

## Concepts Considered

Our clients provided us with three research papers that explored motion data. Some of the concepts in the paper were established as methods to utilize for this project and are detailed in the following sections.

1. MotionGPT (<https://motion-gpt.github.io/>)

MotionGPT is a specialized Large Language Model (LLM) that bridges the gap between motion and text, enabling the generation of human movements from text descriptions, and vice versa. In MotionGPT, tokens represent specific elements or features of human motion, similar to how words are used as tokens in an LLM. By encoding motion data into tokens, MotionGPT can effectively process and generate motions just as traditional LLMs handle text. The idea was to utilize the capabilities of MotionGPT to provide text feedback for a series of motion tokens that we would build from the input exercise. However, despite its promising potential, our team encountered challenges while attempting to implement MotionGPT for several reasons. Primarily, we found the documentation for MotionGPT to be lacking, making it difficult to fully grasp its capabilities and integrate it into our workflow. Secondarily, we observed that MotionGPT performed poorly with specific movements, indicating limitations in its ability to generalize across all types of motion data, especially those required in making distinctions between correct and incorrect performances of the same exercise. Lastly, our project primarily involved 2D motion data, which posed significant hurdles in converting it into the 3D motion tokens required by MotionGPT. These obstacles made it impossible for our team to utilize MotionGPT to give text feedback for input exercises.

1. Skeletal Movement Capture Models
   * 1. PoseFormerV2 (<https://github.com/QitaoZhao/PoseFormerV2>)

PoseFormerV2 was a skeletal motion capture model we explored for this project. This model extracts a 32 joint skeleton from a 2D plane (from a video in an AVI or MP4 format) and then transforms this information through a regression head and time-frequency fusion transformer to map the movements onto a 3D plane. This would have made a smooth transition into the 3D MotionGPT motion tokenizer due to the 3D data format.

To limit computation time, PoseFormerV2 selects only the central frames of the uploaded video to analyze the movement. The last step of this model transforms the 32 joint skeleton into 17 joints.

This model was particularly of interest due to its noise reduction and lengthy sequence analysis abilities with a higher accuracy rate than that of PoseFormer.

This model uses the Human3.6M dataset which was not available to the public without requesting permissions from the administrators. Our team did request permission but received no response. Because we did not have access to the dataset that this model uses, we were not able to extract skeletal data with this model.

* + 1. BlazePose（ https://github.com/geaxgx/depthai\_blazepose ）

BlazePose is an advanced posture estimation solution developed by Google, designed for the real-time and accurate detection and tracking of human body postures. It uses deep learning models to identify and track the key points of the human body, thereby analyzing and understanding human movements. The design of BlazePose allows it to run on a variety of devices, including low-power mobile devices and high-performance server platforms, making it suitable for a wide range of application scenarios, such as fitness tracking, augmented reality (AR), and human-computer interaction. One of the major advantages of BlazePose is its balance between real-time performance and accuracy. It employs an efficient neural network architecture that achieves fast processing speeds while maintaining high accuracy. Furthermore, BlazePose is specially optimized for tracking dynamic postures and rapid movements, making it perform exceptionally well in scenarios such as motion analysis and sports training.

The number of key points BlazePose can detect depends on the version of its model. As of my last update (December 2023), BlazePose has two main model versions:

Lite and Full models: These models can detect 33 human body key points. These key points include parts of the body such as the head, shoulders, arms, wrists, hips, knees, and ankles.

Heavy model: This version of the model is designed to provide more detailed pose estimation and can detect 39 key points. In addition to the key points provided by the Lite and Full models, the Heavy model can detect additional key points, offering richer information about body posture.

* + 1. OpenPose

OpenPose is a deep learning-based library used for detecting and tracking key points on the human body, face, and hands in images or videos. In our project, we utilized the body\_25 model variant of OpenPose, which identifies 25 key points on the body in a single frame. However, accessing the necessary model parameter files posed a challenge, as they were too large to be hosted on the project's GitHub repository. We had to locate and download these files from a creator's Dropbox before we could integrate them into our project successfully. Since OpenPose was the only pre-trained joint detection model that the team was able to access, this is what was used to process all the video data in this project.

1. “A-Deep-Learning-Framework-for-Assessing-Physical-Rehabilitation-Exercises"

The model proposed in this framework used in our development and research is the Spatio-Temporal Model. The published model uses Vicon data collected from 10 subjects; the data from these subjects is stored in the UI-PRMD database. Vicon is a system that uses physical motion capture sensors placed on 39 joints on the human body. Vicon captures positional and angular data of the subject as they move. The model used angular data as input. Each motion capture sensor captured an x, y, and z degree of movement for each joint resulting in 117 dimensions, used as input to the model. Because Vicon can capture information at a high rate of speed, each repetition had a large amount of data. 240 timesteps were used for each repetition of an exercise.

The architecture of the model is shown Figure 3 below. The first step in this model is to separate each exercise into 5 parts: the right and left arm, the right and left leg, and the trunk. Then, this data is passed into a temporal pyramid sub-network to split sequences into full length, half length, quarter length, and eighth length sequences. The split sequences pass into two rounds of convolutional layers. Then, the concatenated sequences are passed into four LSTM recurrent layers, before predicting and accuracy score. A diagram of a person running

Description automatically generatedFigure 3: The architecture proposed in ”A-Deep-Learning-Framework-for-Assessing-Physical-Rehabilitation-Exercises"

For our proposed development, we would create a database of videos using skeletal extraction of uploaded videos and alter the model to use this data as input instead of the Vicon data. The architecture would remain the same.

## Concept Selection

The team continued the work initiated by Dr. Vakanski and Dr. Min Xian on their project titled "A-Deep-Learning-Framework-for-Assessing-Physical-Rehabilitation-Exercises." We adopted their Spatio-Temporal training model, modifying it to use positional data from the OpenPose model, which had demonstrated efficacy in a similar project utilizing data collected by Vicon. The team opted against integrating a Large Language Model (LLM) into the exercise feedback process due to concerns regarding the viability of MotionGPT, as discussed later in the document. Instead, the team chose to develop their own deep neural network structure in addition to adapting the existing one. This decision stemmed from the belief that the current Spatio-Temporal structure might have been overly intricate for the task at hand. This is because the dimensionality of the Vicon data was very large (117x240 per exercise) in comparison the the OpenPose data that would be used in this project (50x40 per exercise). The selected architecture utilized LSTM layers from the Keras Python library to construct Recurrent Neural Networks, which are suitable for processing time series data, which served as the model's input. It should be noted that all of the UI-PRMD database of 2000 individual exercise videos was used for out training. These videos were recorded in Idaho Falls and used for "A-Deep-Learning-Framework-for-Assessing-Physical-Rehabilitation-Exercises". It consists of 10 different types of exercises, where each type of exercise has 10 good repetitions recorded by each of 10 subjects, along with 10 bad repetitions recorded by each of the same 10 subjects.

## System Architecture

1. OpenPose (<https://github.com/CMU-Perceptual-Computing-Lab/openpose>)

OpenPose is a human key-point detection open-source project developed by Ginés Hidalgo and others. From this project, we used the pre-trained body\_25 model. This model is a convolutional neural network model with 105 megabytes of parameters. This model takes a square image as input, and returns 26 coordinate pairs (the x and y coordinates of where the model thought each of these 26 human key points are located: Nose, Neck, RShoulder, RElbow, RWrist, LShoulder, LElbow, LWrist, MidHip, RHip, RKnee, RAnkle, LHip, LKnee, LAnkle, REye, LEye, REar, LEar, LBigToe, LSmallToe, LHeel, RBigToe, RSmallToe, RHeel, Background) along with a corresponding confidence level for each predicted coordinate pair. See the image below (Figure 4), where the output of the body\_25 model is plotted over the input image. The team found that this model sometimes struggled to distinguish some human key points from equipment in the background of an image and would regularly return coordinates that were incorrect; hence, care had to be taken to systematically recognize and correct faulty data.

To use the OpenPose model for video processing, a python script was used to divide up a video into frames and run each frame through the body\_25 model. The output of each frame was stored as a 1-dimensional array, and all of these arrays from a single video’s output were combined to make a single 2-dimensional array. This allowed every video processed by OpenPose to be stored conveniently in csv file format.

A person lifting a bar

Description automatically generated

Figure 4: This is a frame of a video analyzed by OpenPose, with the identified joint locations represented by red dots.

1. Data Preprocessing
   1. Smooth out Outliers

As mentioned in the previous section, the body\_25 model from OpenPose frequently produced faulty joint locations, usually because of equipment visible in the background of the frame. These coordinates had to be corrected before input into any other models. To correct these errors, human key points were analyzed individually over time. If any x or y was a statistical outlier relative to its location in the frames immediately preceding or following it, then it was assumed to be a mistake. These erroneous points were corrected to the average of their nearest neighbors (in the temporal dimension). This method of correcting outliers not only fixed blatant errors in the output of OpenPose’s body\_25 model, but also served to smooth out some of the local imprecision that occurred frame to frame (e.g., in one frame, the right knee’s coordinate might have been placed at the top of the right knee, and in the next frame, this coordinate might have been placed toward the bottom of the right knee).

* 1. Interpolate

The smoothed output from OpenPose’s body\_25 model comes with a row of coordinate values for every frame in the input video. Unavoidably, videos of people performing exercises come in many different numbers of frames. They vary based on how long the subject took to perform the exercise, the framerate of the video camera, and how much additional time is on the front and back ends of the video when the subject has not started or is done performing the exercise. This posed a problem, because all of the machine learning model architecture’s that we employed required that input data be of uniform dimensionality. This means that the “frame count” for each individual exercise csv must be identical. To satisfy this constraint, each column of every csv file was piecewise linearly interpolated to 40 values before used as input to any of the other models.

* 1. Normalize data

In order to prepare the data from each CSV file for use in our models, a normalization process was employed to standardize the feature values. This normalization step was crucial for ensuring that the model could effectively learn from the data without being biased by differences in the scale of the features. Specifically for input into our Recurrent Neural Network (RNN) models, each column in the CSV representing the coordinates of body key points was normalized independently. This involved scaling the values of each key point coordinate to fall within a similar range (each coordinate was centered at 0 and had a standard deviation of 1 over time). For the Spatio-Temporal Models, all values in a CSV file were centered on the zero mean and scaled to fit on [0,1]. This difference in normalization technique was possible (and required) because of the differences in model complexity between Spatio-Temporal and the RNN models. Figures 5 and 6 visualize example scaled data.

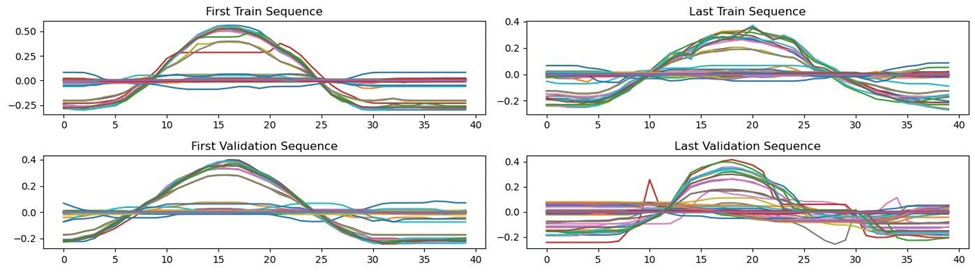


Figure 5: Deep Squat OpenPose data over 40 frames when centered on the zero mean and scaled between –1, and 1.

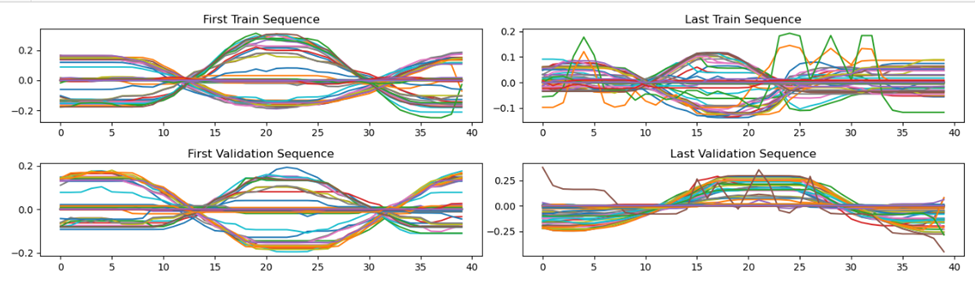


Figure 6: Side Lunge OpenPose data over 40 frames when centered on the zero mean and scaled between –1, and 1.

* 1. Reshape data

The data from OpenPose was extracted to a CSV file, in the format of frames of the video as rows and the x, y, positions of each joint as columns. After smoothing, interpolating, and normalization as explained previously, the data is 40 frames by 50 joint positions (x, y of 25 joints). We then inserted a z point for every joint, totaling the columns to 75. The Spatio-Temporal model takes this information as input with the axes switched. The NumPy swapaxes() function was used on the data to switch rows and columns.

1. Spatio-Temporal Model

The original Spatio-Temporal model uses 117 dimensions from the Vicon angular joints (x, y, z) as input. See Figure 7 below for Vicon joint details. Each repetition contains 240 timesteps. Each sequence is segmented into 5 limbs: the trunk, arms, and legs. The Vicon joints were segmented as follows:

Trunk: Left Clavicle, Right Clavicle, Left Thorax, Right Thorax

Right Arm: Right Shoulder, Right Elbow, Right Radius, Right Wrist, Right Upper Hand, Right Hand

Left Arm: Left Shoulder, Left Elbow, Left Radius, Left Wrist, Left Upper Hand, Left Hand

Right Leg: Right Pelvis, Right Hip, Right Femur, Right Knee, Right Tibia, Right Ankle, Right Foot

Left Leg: Left Pelvis, Left Hip, Left Femur, Left Knee, Left Tibia, Left Ankle, Left Foot

To adjust this model to use OpenPose data as input we segmented the OpenPose joints similar to the Vicon segmentations. OpenPose outputs an x and y positional coordinate for all 25 joints for each frame of video. The OpenPose joint order is detailed in Figure 8 below. We added a z point to every joint to bring the dimensions of input to the model to 75. A typical smartphone records video at about 30 frames per second, and a typical repetition of an exercise lasts 2-3 seconds. We interpolated the full length of the exercise to fit 40 frames per exercise. The segmentation of the OpenPose joints into 5 limbs is detailed below:

Trunk: Left Ear, Right Ear, Left Eye, Right Eye, Nose, Neck, Left Shoulder, Right Shoulder, Middle Hip

Right Arm: Right Shoulder, Right Elbow, Right Wrist

Left Arm: Left Shoulder, Left Elbow, Left Wrist

Right Leg: Right Hip, Right Knee, Right Ankle, Right Heel, Right Big Toe, Right Small Toe

Left Leg: Left Hip, Left Knee, Left Ankle, Left Heel, Left Big Toe, Left Small Toe

The architecture of the OpenPose version of the Spatio-Temporal remained the same. Details of the architecture for this model can be found under the “Concepts Considered” section of this paper.

A testing function was added to the model to predict an accuracy score for a preprocessed video uploaded by the user. The function takes a csv with 40 frames along the columns and 75 dimensions along the rows. This function segments the limbs into the order listed previously, and then predicts an accuracy score using the model.predict() function from the Keras library.

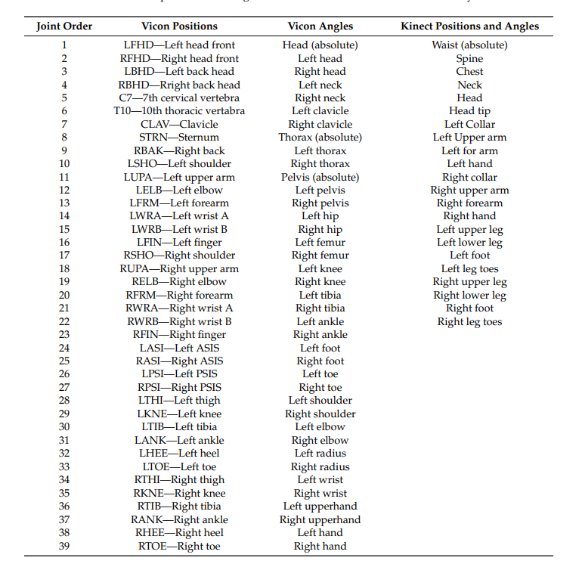


Figure 7: Vicon angular and positional joint order. Figure 8: OpenPose positional joint order.

1. Recurrent Neural Network

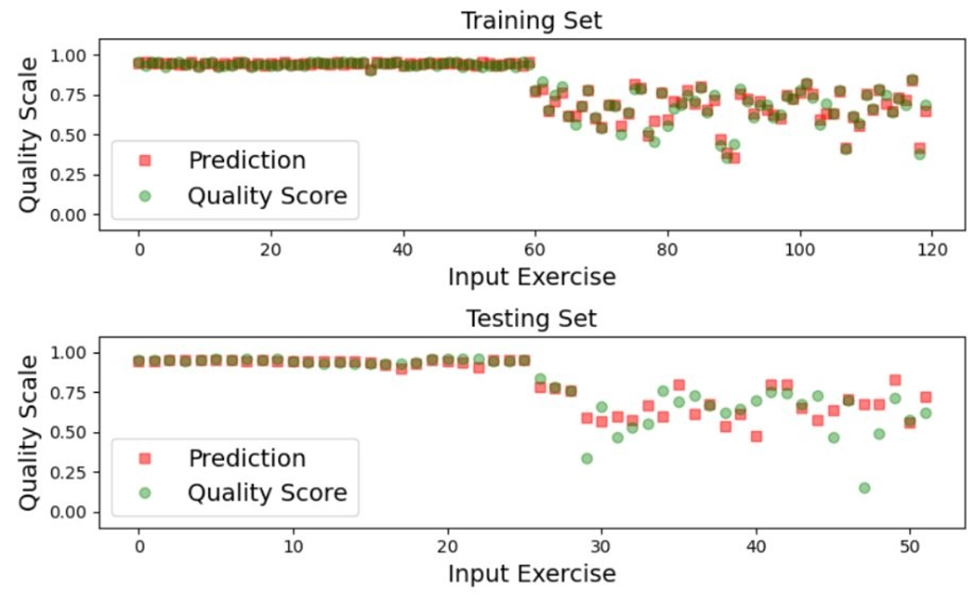
The recurrent neural network (RNN) employed by the team leveraged a Keras LSTM (Long Short-Term Memory) layer followed by a fully connected layer and an output layer tailored for either classification or regression tasks. For the classification task, which aimed to discern between three distinct exercises (deep squat, side lunge, inline lunge), the output layer comprised three nodes utilizing the softmax activation function. This model was trained on a dataset consisting of 600 individual exercise videos from the UI-PRMD dataset, with validation conducted on approximately 100 exercises recorded by team members and their acquaintances. Additionally, the team developed similar models for each of the three exercise types to provide a quality score for each exercise. These models featured larger LSTM layers, fully connected layers, and a single output node with the sigmoid activation function. Training data for the quality score model included all UI-PRMD videos of the respective exercise type (correct movement quality score = 1, incorrect movement quality score = 0.6), supplemented by some UI-PRMD videos of other exercises (quality score = 0), with immediate testing set validation being performed on a subset of this training dataset.

## Design Evaluation

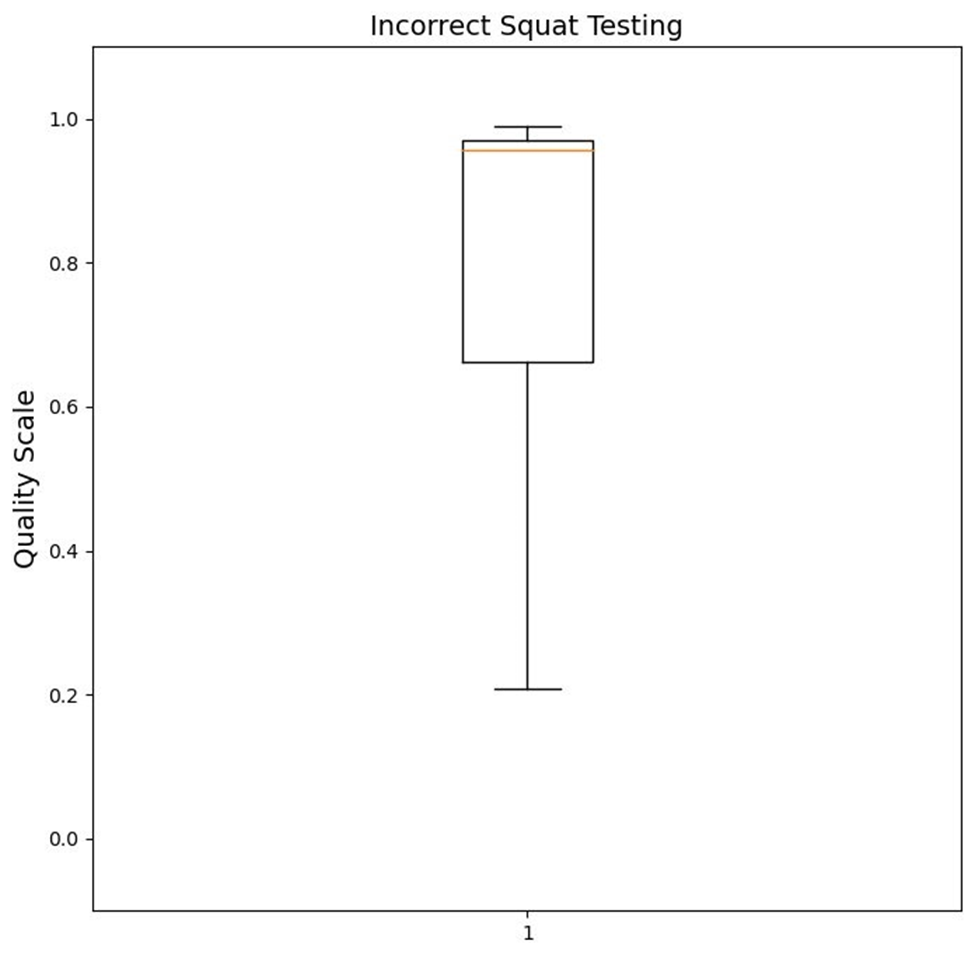
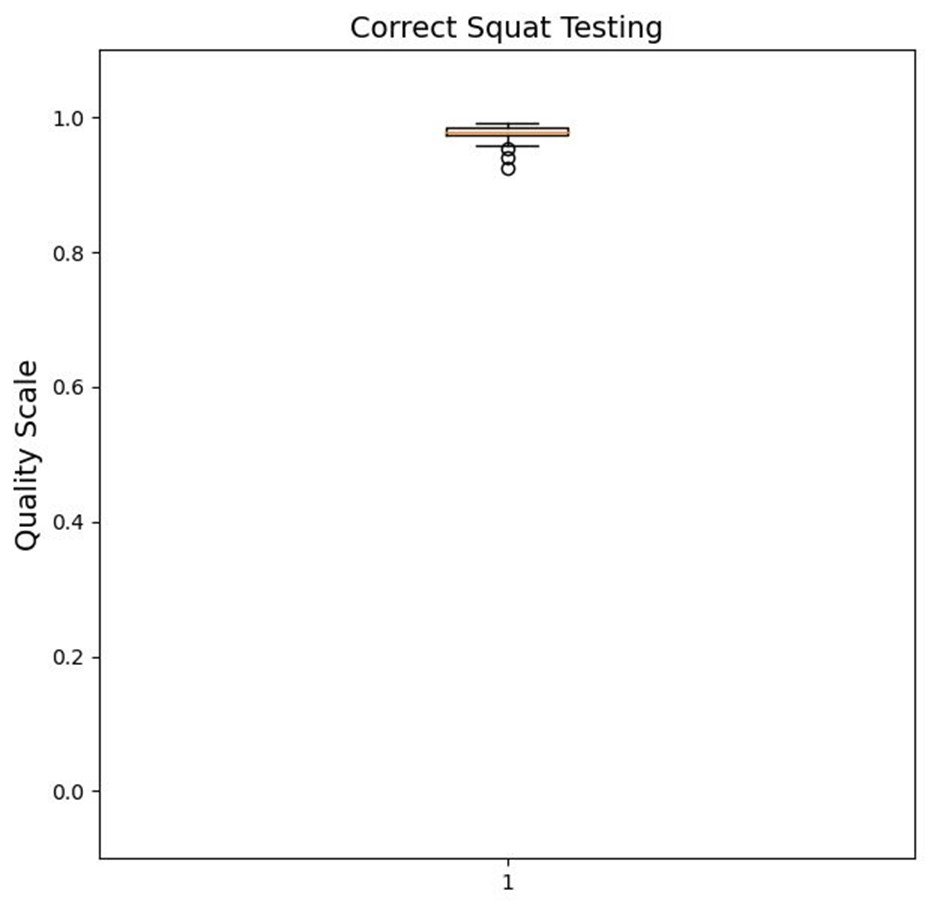
Our design validation and results plan are shown in the appendix below. There are detailed tests, results and recommendations for each of our project requirements. Validation for each of the models is detailed in the following sections. We used python plot libraries (matplotlib.pyplot) to analyze sequences, training loss/validation loss, and mean absolute deviation and RMS.

To test the developed model, the team collected a dataset of self-recorded single exercise smartphone videos. These videos were uploaded to a personal machine for analysis. Utilizing the OpenPose framework in conjunction with a smoothing algorithm, skeletal movements were extracted from the video recordings. Subsequently, the extracted data underwent a series of preprocessing steps, including interpolation, centering, scaling, and reordering, to ensure alignment with the format required for input into the deep learning model. With the prepared data, the team then fed it into the model to produce prediction quality scores for each exercise video. To assess the accuracy of these scores, they were compared against a reasonable range established for the quality score of each respective exercise. This validation step provided insights into the model's effectiveness in evaluating exercise quality.

1. Spatio-Temporal OpenPose Model Validation

The Mean Absolute Deviation between the model predicted scores and the actual quality scores in the testing set of the deep squat exercise was 0.06007. The RMS (root-mean-square) deviation was 0.109300. Compared to the mean absolute deviation and rms deviation of the original Spatio-Temporal proposed model (0.009540543978214275, and 0.014649246638018216 respectively), we can see that this model is less accurate. As detailed in the training and testing set diagram below (Figure 9), we can see that the model predicts correct repetitions of the exercise well but tends to predict a high accuracy score for incorrect repetitions of the exercise. Figure 9: Training and testing set quality scores vs machine prediction for the Spatio-Temporal model with the deep squat exercise.

We can see this further proved by mapping the distributions of the accuracy predictions of individual uploaded correct and incorrect repetitions for testing.

Figure 10: Boxplot of correct squat repetitions. Figure 11: Boxplot of incorrect squat repetitions.

The figure on the left shows the distributions of 22 correct deep squat videos when tested against this model. All the videos scored high accuracy, which is what the model should predict. The figure on the right shows the distributions of 22 incorrect repetitions of a deep squat, and the median is still high. This model has a high false positive rate when analyzing incorrect repetitions. Out of 22 incorrect repetitions, only 7 of them received an accurate score.

Validation for the model's training and testing sets when trained on the inline lunge, side lunge, and testing of the exercises are detailed in the figures (12 & 13) below.

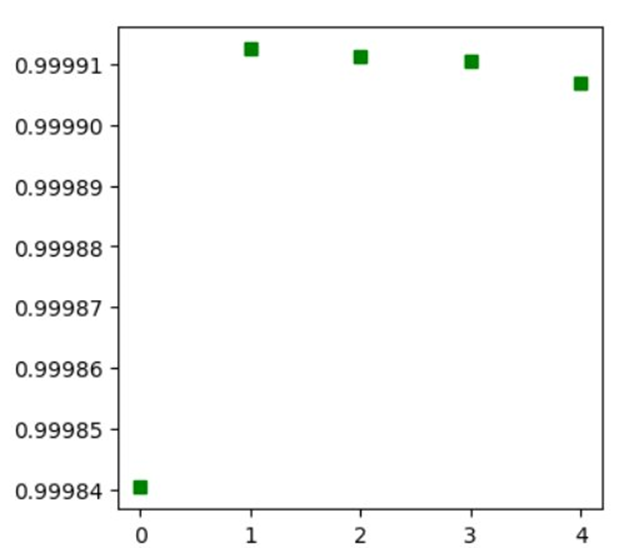


Figure 12: Testing input of 5 correct repetitions of a side lunge exercise on the Spatio-Temporal model using OpenPose data.

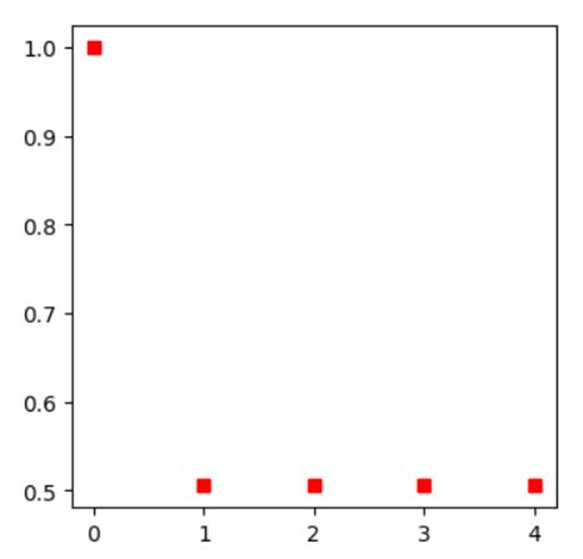


Figure 13: Testing input of 5 incorrect repetitions of a side lunge exercise on the Spatio-Temporal model using OpenPose data.

1. Recurrent Neural Network Model Validation

The recurrent neural network model was more accurate than the Spatio-Temporal model, especially when predicting quality scores for incorrect repetitions for the exercises. We validated the recurrent neural network models by checking how well the model performed on new data. We found that this model was better at generalizing what was learned in the training process (from the UI-PRMD dataset) to new data (the smartphone video dataset of team members and their acquaintances performing exercises). This can be seen in the boxplots below (Figure 14): The correct movements are all scored higher than the incorrect movements for each new subject, and exercises that are not even the specified exercise (lunges instead of deep squat) are scored the lowest. For this reason, the Recurrent Neural Network is the recommended model when using movement data extracted by OpenPose.

A diagram of squat quality distribution

Description automatically generated

Figure 14: Validation data output distributions of the Deep Squat RNN. Note that each boxplot represents the output scores for about 10-20 individual exercise videos.

The recurrent neural network structure also exhibited strong performance when classifying exercise videos as deep squat, inline lunge, or side lunge. The dataset used to evaluate the classification was the same one used for the recurrent neural network that produces quality scores for exercise (Figure 15). This model was better at classifying correct versions of each exercise (>80% accuracy) than at correcting incorrect versions of each exercise (>65% accuracy). This result is intuitive, because there is more variation in the movements of an incorrect exercise than in a correct one.

A screenshot of a computer code

Description automatically generatedFigure 15: Validation data output of the Classification RNN. Each predicted class label is compared with the true label, and the percent accuracy is provided for each subset of the validation dataset.

## Future Work

Below are recommendations made by our team for future work on our research and development.

1. Larger Database

All machine learning models in this report used the UI-PRMD database for training and testing. This dataset contains 10 subjects. Each subject completed 20 repetitions (10 correct and 10 incorrect) for 10 different exercises. This means that the model was only trained on, at most 100 correct repetitions of an exercise and 100 incorrect repetitions of an exercise. Some of the repetitions were unusable in the dataset due to cuts in the video, or a bad representative repetition. Additionally, the videos in the UI-PRMD database were recorded by subjects with the same camera, same camera angle and same background.

For better generalizability and accuracy, our recommendation would be to create a larger database of the selected exercises. Additional data would allow the model to have a longer training and larger testing set. Including videos with a variety of backgrounds and camera angles would allow the model to produce accurate feedback scores if a user uploaded a video without the same angle as the models were trained on.

1. Additional exercises

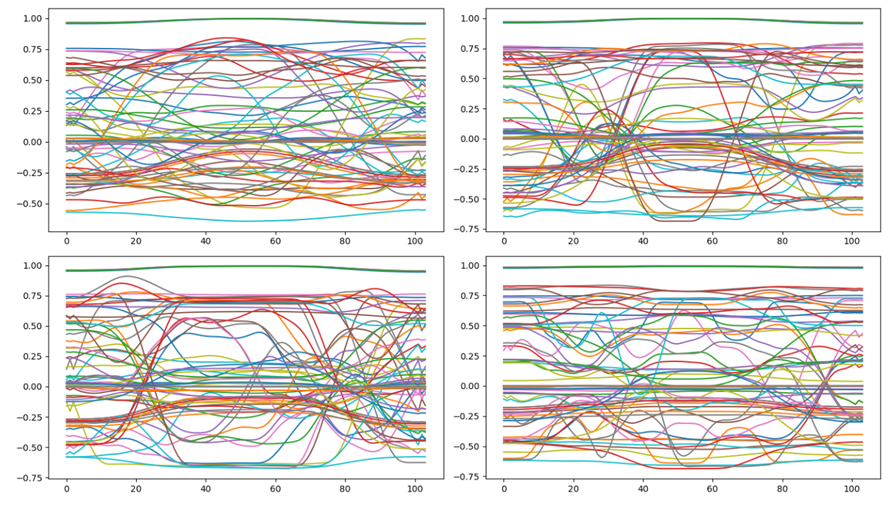
Our models were trained on the deep squat, inline lunge, and side lunge exercises. The UI-PRMD database includes 10 different exercises. Future expanse could include training these models with the other exercises in the database.

The framework of our current models could also expand to other forms of movement. Some examples recommended to us throughout our project have been golf swing form, alpine skiing movements, and rock-climbing beta. It could be utilized with any physical movement. Because the goal is to analyze movements from a smartphone device, movements that move along the x or y axis and hardly use the z axis would be ideal.

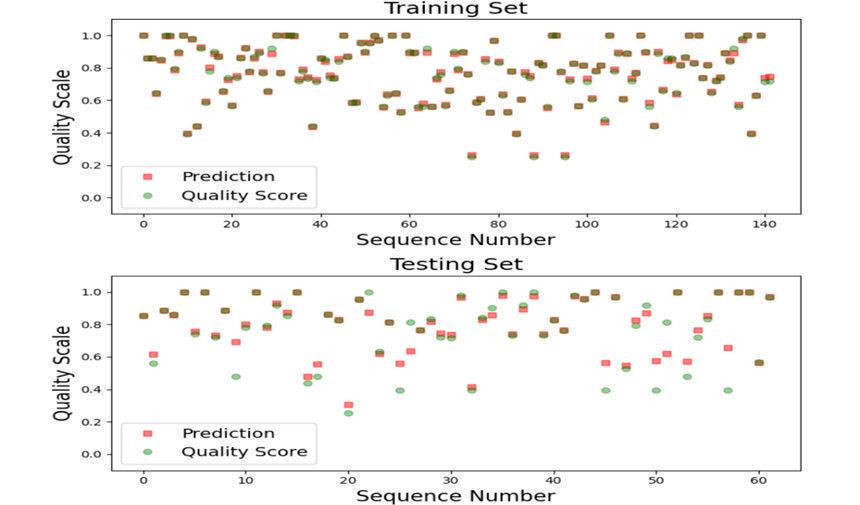
1. Alternative Skeletal Extraction Methods

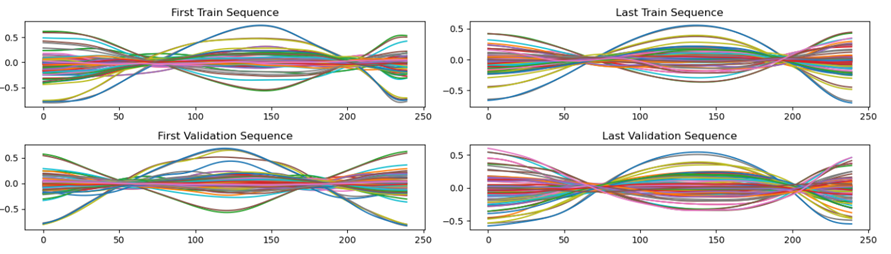
We used the OpenPose body\_25 model to extract skeletal movements from mp4 or avi videos. This extracts 25 body joints (x, and y coordinates). OpenPose had difficulties recognizing background equipment as joints. Other skeletal extraction models that have higher number/increased detail when determining joints could result in a higher accuracy.

4. Compare the accuracy of quality scores from skeletal data obtained through different sources: OpenPose, Vicon, Kinect

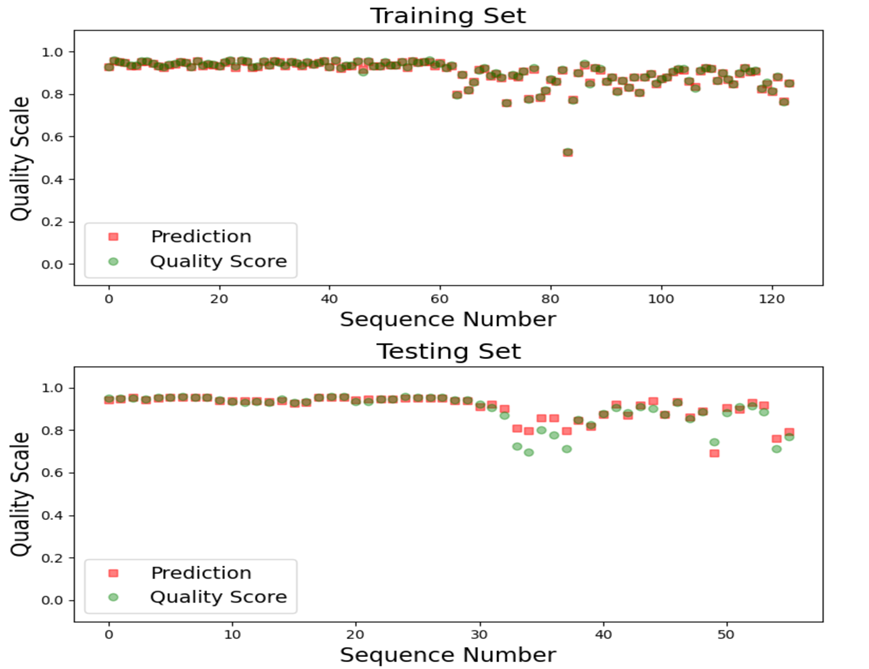


The model's predictions on the training and validation sets.



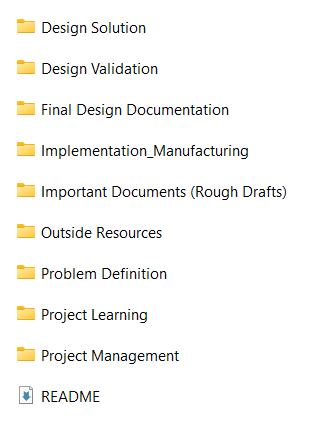


The model's predictions on the training and validation sets.



By comparing the Kinect and Vicon models, we can see that the average quality score for deep squats on Kinect is 0.8018419, while the average quality score on Vicon is 0.92220706. After comparison, Vicon outperforms Kinect. The training time for Vicon is approximately 5 minutes, while the cost for Kinect is about 4 minutes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Design Validation Plan & Results (DVP&R) |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  | Requirement | Test | Test Subject | Target Date | Result | Recommendation |
|  | Software can extract skeletal data from a video. (OpenPose, BlasePose, PoseFormerV2, Video Pose 3d) | Submit a mp4 video with an exercise being performed, and verify that a video output that matches the input is generated. | A video of a deep squat from One Drive | 11/10/2023 | 10/10 - Was able to extract skeletal information from own photos (all 25 joints) | Not PoseFormerV2 or VideoPose3D(No Access) |
|  | Software can condense output from skeletal video to a small datafile. | Software can generate a file that represents joint locations over time. | Skeletal data video | 11/20/2023 | Can output points on joints over frame to an avi and a CSV video | Output to a CSV file |
|  | Applying data preprocessing to the deep squat video exercise | Be able to align all joints correctly on the z axis from the skeletal data extracted from the library we select for this project. | Skeletal data of a deep squat | 11/30/2023 | Able to extract x, and y joints. Z points will either be the confidence interval or manually set to zero. |  |
|  | Tokenizing the deep squat skeletal data using motionGPT | Submit the skeletal data that has been preprocessed from the method above, and have it generate a motion graph | Skeletal data of a deep squat | 12/21/2023 | 1/18/2024 - downloaded some sample npy data from MotionGPT but is pretty unreadable/no documentation | Cannot figure out how to input 2D joint information to a 3D angular information system that MotionGPT uses |
|  | Our model can identify the deep squat exercise. | Submit processed datafiles from videos of people performing a deep squat. | Videos of deep squat exercises (Deep Squat vs Standing Shoulder Abduction) | 3/27/2024 | Simple lstm recurrent NN model can distinguish a deep squat from another exercise but still has issues with determining a good and bad squat | Test with SpatioTemporal model |
|  | Our Program can run within a reasonable amount of time (Under 30 minutes) | Time our program and see how long it takes to preprocess and ouput a prediction score for a deep squat | Video of an exercise (Measure with a stopwatch) | 4/8/2024 | Video of bad squat is processed on a CPU in (18:32) | If higher speed is needed we could process the video on the GPU server. |
|  | The model can differentiate good exercises from bad ones, using newly generated data with smartphone | Submit videos to system of Molly doing a squat | Videos of deep squat exercises (3 bad and 2 good) | 4/1/2024 | Poor accuracy ranges: [0.09964518],[0.01250687],[0.0168968 ],[0.02202759,[0.01254246] | Augment data source by our own videos |
|  | Prepare OpenPose Data for NN (data preprocessing to fit the SpatioTemporal model) | Submit a video of a squat with good form -> no constructive feedback | Videos of exercises | 2/20/2024 | We have two methods of preprocessing the information to provide to the neural network. This consists of running a video through the OpenPose software (openpose.py), to get x, y skeletal joint positions, then smoothing the data using an algorithm in (smoothAll.py) and finally interpolating the columns to extract the same number of frames per repetition (interpolate.py). Additional centering/reshaping the data to fit into the spatiotemporal network can be found in the "SingleEpisodePreprocess.ipynb" script. | Write a script that runs all these scripts to streamline the process. |
|  | Compare the quality scores by the NN models based on poses obtained by OpenPose, Kinect, and Vicon systems. | Submit a video of a squat -> labels from 3 different methods | Videos of exercises | 3/1/2024 | The training loss and validation loss of Vicon system is lower than these of Kinect system | If we need shorter training time, we recommend using Kinect system, otherwise we recommend using Vicon system. |
|  | Apply the existing models for rehabilitation evaluation developed to predict quality scores for squat, based on the estimated poses from the videos with OpenPose. | Submit a video of a squat -> labels from OpenPose | Videos of exercises | 3/1/2024 | Good Squat Accuracy - 0.9186... Bad Squat Accuracy - 0.8311407 Since model training is inconsistent we would like to do further analysis of videos to see if that is an underlying issue. | Model Training is very inconsistent so the prediction labels change every time the model is retrained even on the same data. Recommend more data for training and a better labeling process. |
|  | Model Training Time (Under 20 minutes) | Run the model with dataset to see how long training takes. | Submit information for the model to be trained on (time with python time function) | 4/8/2024 | Training time: 0:03:50.833147 | If faster training is needed, can use a GPU server instead of CPU but this had a relatively fast run time. |
|  | Extra Design Testing | Low RMS, Low Training Loss, Low Mean Absoulte Deviation | Print these statistics when training the models | 4/9/2024 | RMS: 0.10607357120918924 MAD: 0.0726833657075426 Training Loss: 0.5364628434181213 Validation Loss: 0.5352079272270203 | Reference to original spatiotemporal for Vicon model: Training time: 0:17:52.663051 Training loss 0.2981928517260859  Validation loss 0.3056857282561915 Mean absolute deviation: 0.0726833657075426  RMS deviation: 0.10607357120918924 |



An overview of the organization of the OneDrive folder where the files are stored.

## Citations

@ARTICLE{Liao2020,

title={A Deep Learning Framework for Assessing Physical Rehabilitation Exercises},

author={Liao, Y. and Vakanski, A. and Xian, M.},

journal={IEEE Transactions on Neural Systems and Rehabilitation Engineering},

year={2020},

month={Feb.},

volume={28},

number={2}

pages={468-477},

}

@article{8765346,

author = {Z. {Cao} and G. {Hidalgo Martinez} and T. {Simon} and S. {Wei} and Y. A. {Sheikh}},

journal = {IEEE Transactions on Pattern Analysis and Machine Intelligence},

title = {OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields},

year = {2019}

}

@inproceedings{simon2017hand,

author = {Tomas Simon and Hanbyul Joo and Iain Matthews and Yaser Sheikh},

booktitle = {CVPR},

title = {Hand Keypoint Detection in Single Images using Multiview Bootstrapping},

year = {2017}

}

@inproceedings{cao2017realtime,

author = {Zhe Cao and Tomas Simon and Shih-En Wei and Yaser Sheikh},

booktitle = {CVPR},

title = {Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields},

year = {2017}

}

@inproceedings{wei2016cpm,

author = {Shih-En Wei and Varun Ramakrishna and Takeo Kanade and Yaser Sheikh},

booktitle = {CVPR},

title = {Convolutional pose machines},

year = {2016}

}