PT Smart Assistants: Running Form Sensor

Team Name: PT Smart Assistants

Team Members and Roles:

1. Clyde Bango (Project Lead)

Abstract:

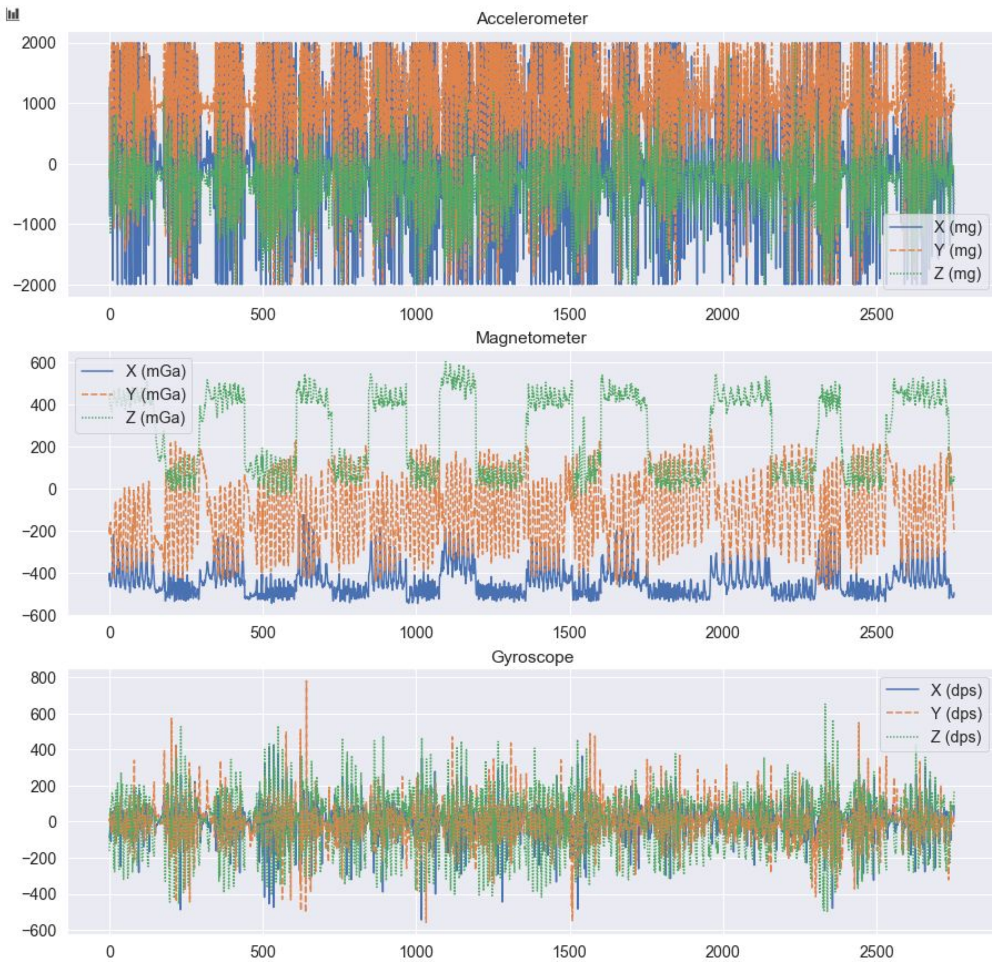
Poor running form per physical therapy specifications can lead to problems of painful muscle and joints, like osteoarthritis of the knee. When athletes maintain the right form while running, they can prevent many injuries, and reduce muscle pain. Professional and other seasoned athletes go through rigorous strength and conditioning with physical therapists. They use highly advanced wearable devices to monitor their running form, collecting data that can be analysed by biomechanic specialists for feedback and tracking progress. It is highly recommended for anyone who can afford a physical therapist to consult with the provider, not just after an injury has occurred but for feedback and conditioning. Casual runners do not have the benefit of professional devices and real-time feedback on their running form. A wearable device can help track stats and provide feedback to runners. Several devices have been introduced to the market, including smart watches, smart shoes, clip-on sensors that can be attached to shoes, belts, clothes, and even Bluetooth headphones equipped with lightweight motion sensors to analyse gait.

This project explored the feasibility of a connectable sensor node from the SensorTile development kit (STMicroelectronics) to prototype a Bluetooth-smart and Machine Learning sensor fusion system that can predict good running form vs bad running form categories. The results provided a proof of concept with regards to hardware setup, data collection and processing, and using a deep learning approach to classify data learned features of optimal versus suboptimal running form.

Data Collection:

The sensor node was attached to the anteromedial knee (inside-front part of the knee) to monitor running form, tracking statistics in 3 dimensional axis and remotely sending the data to the ST BLE mobile app via bluetooth. In a more robust setup, multiple sensors can be used to record data from the different points. Part of the project motivation is to show the most minimal number of sensors that can capture nuances of ‘good’ vs ‘bad’’ running form. The positioning of the sensor was strategic in order to capture the speed, angles and direction of the legs while running; accounting for factors like stride, foot strike, cadence and q-angle. Data was collected for accelerometer, gyroscope and magnetometer features. About 70 optimal running samples and 50 anomaly running samples were used for the scope of this experiment. Samples were measured at 10 second intervals. A physical therapist consultant provided the feedback on running form categories and kept track of sample times. Videos are provided in the github folder to show the data collection live demo.

```
plot_features(data, features)
```



Data Processing:

Custom functions were built in python to load and pre-process the data for modeling. Input files were in csv format, sent by email and downloaded to a local machine. The jupyter notebook "load_data.ipynb" shows the workflow of creating and testing the functions. The accompanying data_loader.ipynb contains the defined functions that can be imported to other notebooks for routine data loading and processing. Data can be concatenated for the three features and sliced into the 10 second intervals for each sample. Visualizations and summary statistics help to understand the data and validate the logic of slicing the samples. To even the time steps, the data was resampled at

0.05 seconds. Edges when the runner was transitioning to the next 10 second interval were trimmed since the sensor detected a resting signal than running form.

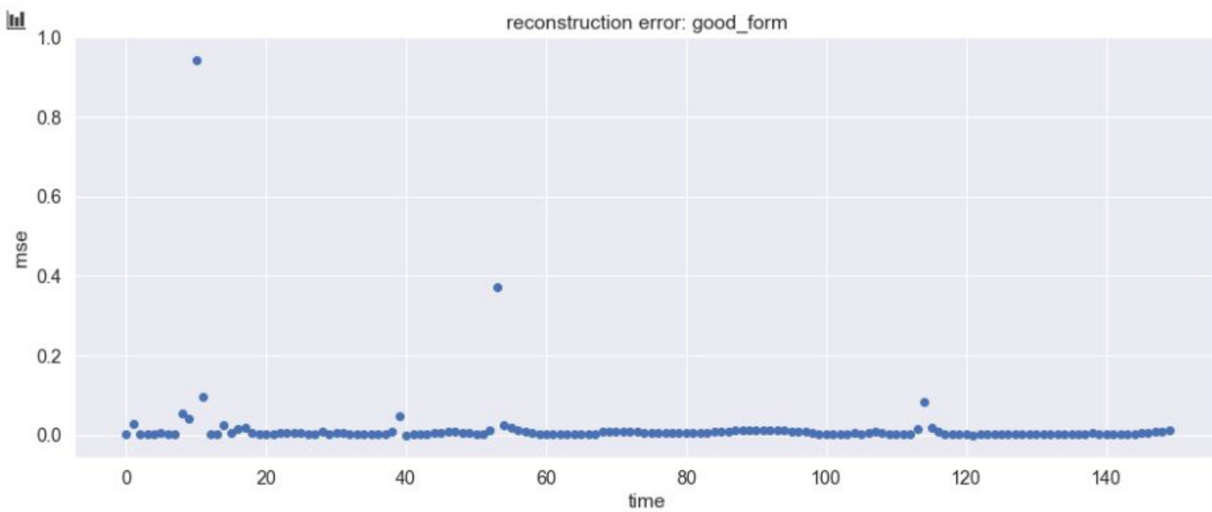
Modeling:

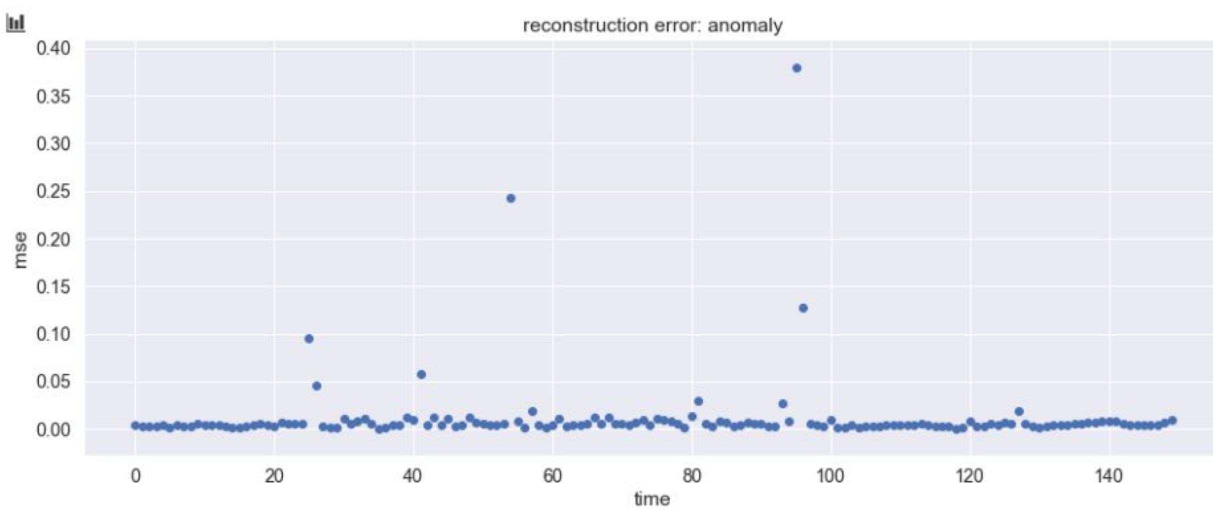
To complement the various signal processing and feature extraction methods we explored in the course, this project considered the use of deep learning approaches to perform automatic feature learning. Deep learning systems can discover important representation features that may be used to classify high dimensional, non-parametric, often large datasets. Autoencoders provide an unsupervised learning technique via encoding representations of the input data and reconstructing aspects of the input signal using the decoder phase of the model. The input features are forced through a bottleneck, forcing the model to compress knowledge of the underlying structure in order to succeed in the reconstruction task. By using time distributed layers in the autoencoder, we can track the timepoints and evaluate the reconstruction loss across the time domain. The model architecture is shown below.

```
# summary of the layers in the autencoder
autoencoder = get_autoencoder(stacked_data)
autoencoder.summary()
```

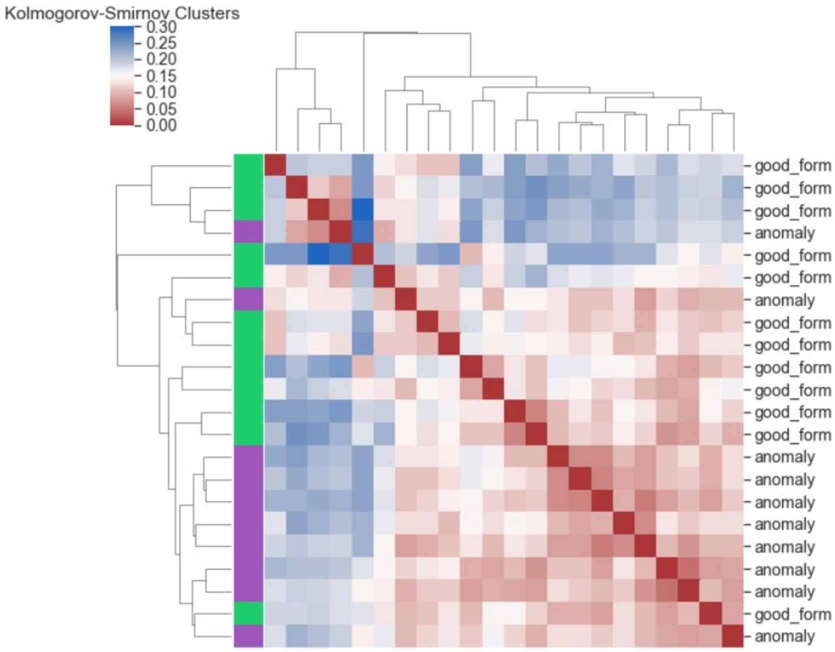
```
Model: "functional_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 150, 9)]          0
_________________________________________________________________
time_distributed_36 (TimeDis (None, 150, 200)          2000
_________________________________________________________________
time_distributed_37 (TimeDis (None, 150, 50)           10050
_________________________________________________________________
time_distributed_38 (TimeDis (None, 150, 10)           510
_________________________________________________________________
time_distributed_39 (TimeDis (None, 150, 50)           550
_________________________________________________________________
time_distributed_40 (TimeDis (None, 150, 200)          10200
_________________________________________________________________
time_distributed_41 (TimeDis (None, 150, 9)            1809
=================================================================
Total params: 25,119
Trainable params: 25,119
Non-trainable params: 0
_________________________________________________________________
```

The input shape of 150 x 9 goes through compressions and decompressions, and finally back to 150 x 9. The output are predictions at each of the 150 timepoints. These predicted values are used to calculate the mean squared error against the original true values. This setup effectively performs dimensionality reduction since we can compare the linear patterns of rmse values, representing how much certain features are prioritized in reconstructing the different classes of running form. Examples of two sample are shown below for anomaly vs good form:

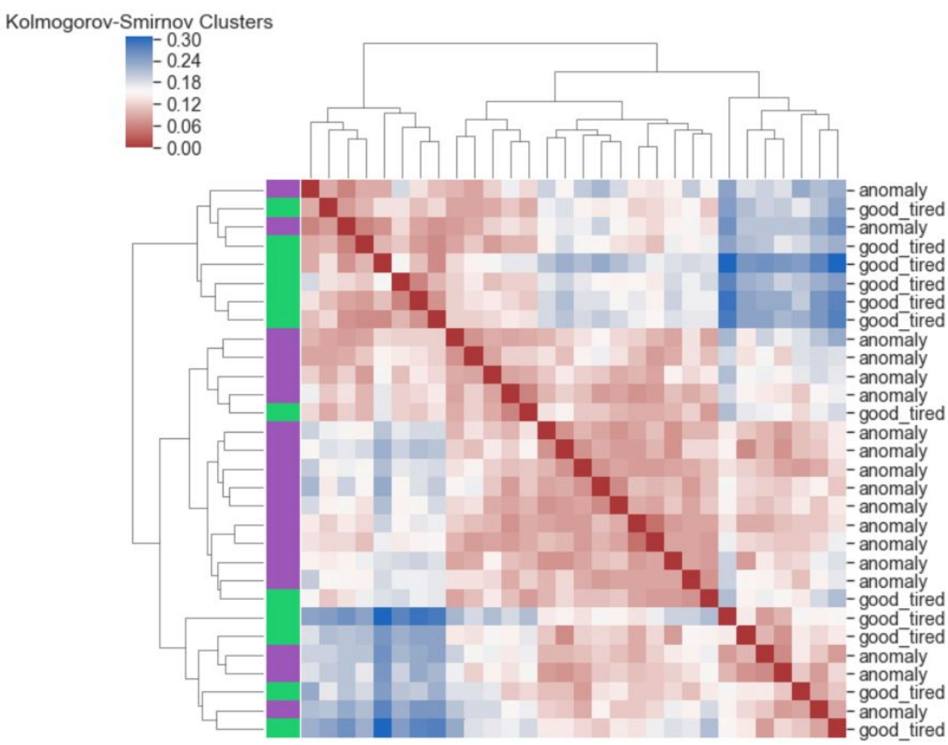




To compare the two classes, mse correlation matrices were calculated for the test data. Since the distribution is non-parametric, Pearson correlation did not perform well. Classes were clustered more effectively using the Kolmogorov-Smirnov test values. The results of hierarchical clustering are shown below:



The above heatmap shows the clustering of good form versus anomaly running form on 22 test samples. A second test set was repeated below, considering good running form when the subject was tired.



Discussion:

The results provide proof of concept that sensor data logged from runners can be clustered using representation learned features to classify running form. A deep learning approach was used for automatic feature extraction, and Kolmogorov-Smirnov test used to analyze the reduced dimensions.

There is signal in the data, and a predictive model can be built to provide feedback in real time. Additional sensors can be added to increase data collection points on both legs and upper body, however with the ultimate goal of using as minimal number of sensors as possible. With more time and team members, a GUI application can be built to provide real time feedback to runners.