Activity Classification and Cadence Estimation Through Smartphone Sensory Data

Group 71: Austin Dickerson[2729612], Georgios Xanthopolous[2755088]

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV
Amsterdam, The Netherlands
https://vu.nl/nl

1 Introduction

This report covers the process by which team 71 conducted their project in Machine Learning for the Quantified Self. The project is centrally aimed at distinguishing user activities, based on data derived from a user's phone accelerometer. After classifying the activity into one of the following categories: Sitting, Walking, Running, Cycling, the associated steps or revolutions per minute are closely tracked. This function is meant to supplement the fitness related data that users collect themselves using various fitness tools. Many people use step tracking tools to analyze their day. Moreover, companies like Technogym are building profile recognition into their machines and keeping databases on users [1]. They send users reports about their activities breakdown and estimated calories burned. We want to bring a holistic tracking method into people's phones so they can make better decisions for their health. Our tracking system goal is unique because it does not just look at pre-designated activity periods. We observe a person's day in sedentary, semi active, and very active points. Users will be able to get a summary of their day that includes minutes walking, running, and cycling as well as the number of steps (or revolutions of the pedal) completed in each activity. This gives people a sense of accomplishment or a reality check if their goals were not reached.

This tool also has a deeper function for performance minded users. Because our model makes continuous predictions for step cadence, a specific activity can have both an average and dynamic step rate. Users can look at their step profile through an activity such as a run on a familiar course to see if their step rate drops too low or gets too high in a particular area or terrain type. Similarly, in cycling, an athlete's efficiency is impacted by cadence [7]. This means a cyclist could look at their logged rides and see where their cadence deviates from the optimal and keep this in mind on later runs of the course.

The objectives for our tool are achieved through the deployment of a mix of classification and regression algorithms. Once the activity in question is accurately identified, the next step is the application of an algorithm pertinent to the activity, to deduce the user's approximate cadence. To accurately determine this cadence, our approach hinges on the recurring cycle of movement during activities such as walking, running, and cycling. This repetitive motion enables us to link a frequency to the acceleration data from the user's phone. The implementation of our proposed system carries extensive benefits, especially in domains such as healthcare, where monitoring and improving patient mobility is essential. By accurately assessing an individual's movement cadence, healthcare providers can gain valuable insights into their patients' physical status, aiding in injury prevention, rehabilitation programs, and general well being. Furthermore, this system opens avenues for improving sports performance by tracking and analyzing athletes' movements, therefore allowing for personalized training regimes. Its relevance also extends to public health, where it can aid in developing and assessing interventions to promote physical activity.

This report is structured as follows: In Section 2, we provide an exhaustive examination of the data types engaged and conduct a comprehensive exploratory data analysis of the collected datasets. Section 3 presents the methods for outlier detection and missing value imputation that were applied. In Section 4, we deliver a detailed overview of the feature engineering process, establishing the groundwork for subsequent analyses. Following this, in Section 5, we articulate the experimental designs and intricately dissect the characteristics of the models utilized. A comparative analysis of the results garnered from various machine learning methodologies is conducted in Section 6. In Section 7, we outline the prospective steps needed to develop a market-ready product. The report culminates in Section 8 with a reflective discussion of the entire research process, which includes insights into potential modifications the research team would consider if given the opportunity to revisit the project.

2 Data description

For the dataset, we chose from a selection of sensory data available through Phyphox, a phone application intended for physics experiments. We made an assessment of each type of data available before settling on Acceleration, Gyroscope, and Linear Acceleration. Magnetic data was not selected, because we are not concerned with how the user is oriented relative to the earth (in terms of location or direction).

The accelerometer is a Micro-Electro-Mechanical System, a sensor with a microscopic mass suspended by springs, which can detects shifts in inertia on the phone [2]. The linear acceleration uses the same tool while excluding the force of gravity. These two readings are strongly correlated, so the information was later assessed through Principal Component Analysis.

The gyroscope works by detecting rotational acceleration. Inside the gyroscope is a tuning fork, which is made to vibrate at a specific frequency [4]. When the tuning fork experiences angular acceleration or rotation on an axis perpendicular to the fork's vibration plane, the Coreolis forces shift the movement of the fork, which is interpreted by into radians per second by the sensor [6].

Each sensor has three readings, for the 'x', 'y', and 'z' axes of the phone. 'x' corresponds to the horizontal plane when looking directly at the phone, 'y' corresponds to the vertical plane, and 'z' to the depth plane. For the purpose of data gathering, the phone was always place in a front pants pocket. The phone sat upside down with the face pointing away from the user. This helped create consistency in the data.

The sensor readings for this data took frequency into account. To select an appropriate frequency, there must be a balance between a fine granularity and conservation of memory. After first sampling at max frequency, each sample came out to tens, if not hundreds, of thousands of rows in a .csv format. This prompted a shift to a 20Hz sampling frequency, as 50ms between original timestamps. That formed the base granularity for later analysis and was compared to higher granularities, where the multiple observations are then averaged per new timestep.

The data was organized in columns corresponding to each measurement, with timestamps in seconds. Table 1 illustrates the formatting for the original data.

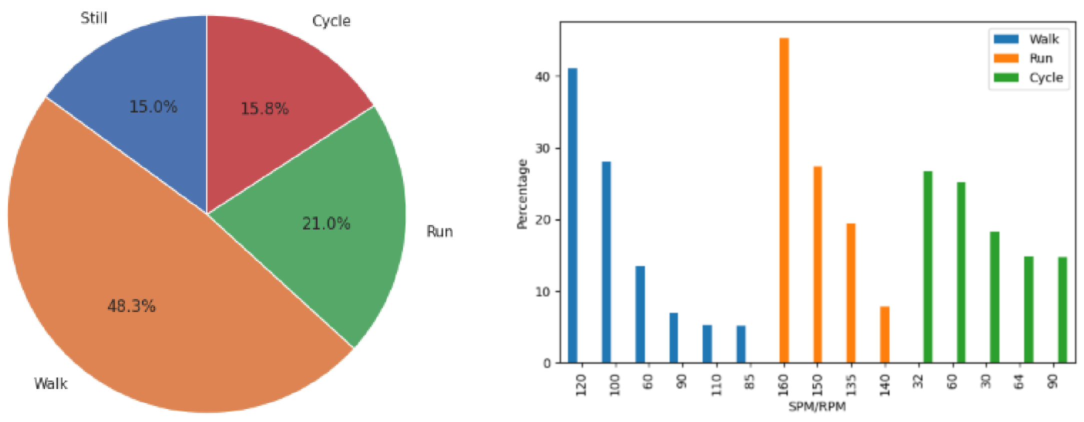
Table 1: Presentation of the raw dataset

Index	Time (s)	x (Acceleration m/s^2)	...	z (Linear Acceleration m/s^2)	Still	...	Cycle	SPM/RPM
0	0.05	3.385545	3.518347	8.998210	0	0	1	64
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15715	385.05	3.385545	3.518347	8.998210	0	1	0	90
15716	385.10	3.294618	3.489633	7.820343	0	1	0	90

2.1 Exploratory Data Analysis

In our Exploratory Data Analysis (EDA) phase, we extensively examined the 2-hour dataset of physical activity tracking, encompassing 144,774 records and four distinct activities: still, walking, running, and cycling. The dataset records were labeled with the activity being performed at each timestamp and an approximation of steps per minute (SPM) for walking/running or rotations per minute (RPM) for cycling. We aimed to familiarize ourselves with the data's nature and distribution, identify potential anomalies, and uncover the underlying patterns that could aid the subsequent modeling phase.

The overall composition of the dataset revealed a class imbalance, with nearly half of the data (48.3%) corresponding to the ‘Walking’ activity. The rest of the records were almost evenly distributed between ‘Running’ (21%) and ‘Cycling’ (15.8%), with a smaller portion designated as ‘Still’ (15.0%). Despite this imbalance, the dataset provided a good variation of activities at different intensity levels, as evidenced by the variety of SPM/RPM values associated with each activity type. An illustration of the composition of the datasets based on the activity type and the SPM/RPM is provided in Figure 1.



(a) Distribution of Activities (b) SPM/RPM Distribution by Activity Type (%)

Fig. 1: Visualization the distribution of activities and cadence (SPM/RPM) in the dataset.

The time-series visual representation of our dataset, as depicted in Figure 2, conveyed notable patterns among different activities. It was evident that sensor readings’ magnitude amplified with escalating SPM or RPM values, reflecting the anticipation that heightened physical exertion would entail more frequent movements and thereby yield larger sensor readings. Particularly, ‘Running’ exhibited a remarkable fluctuation in sensor readings, underlining the high variability inherent to running patterns, while ’Cycling’ presented a more homogeneous pattern in line with its uniform movement nature. Interestingly, these variances extend beyond mere frequency oscillation, hinting at alternative predictive approaches. For instance, the range of values and the standard deviation for acceleration were distinctly affected by the activity type, suggesting that data distribution metrics could prove beneficial in classifying activities. This notion is further supported by the distinctly identifiable distributions of acceleration across ‘Walking’, ‘Running’, and ‘Cycling’ activities, underlining the utility of statistical approaches in activity prediction.

Furthermore, according to the statistical summary of the sensor data provided in the Table 2. The mean and standard deviation values for both accelerometer and gyroscope readings indicated the high variability in our data, which is inherent to the diverse range of movement patterns and intensities produced by different physical activities. Also, the presence of extreme values in our data, such as spikes due to taking the phone out of our pockets between activities, highlighted the richness of our dataset. These outliers, despite appearing as anomalies, are genuine reflections of the variability in our data, contributing to our model’s learning scope and hence its generalization capabilities.

2.2 Granularity

Looking at the filtered data at different granularities, we try to discern which level of detail is most useful for later analysis. We want a level of detail that clearly shows a frequency of acceleration

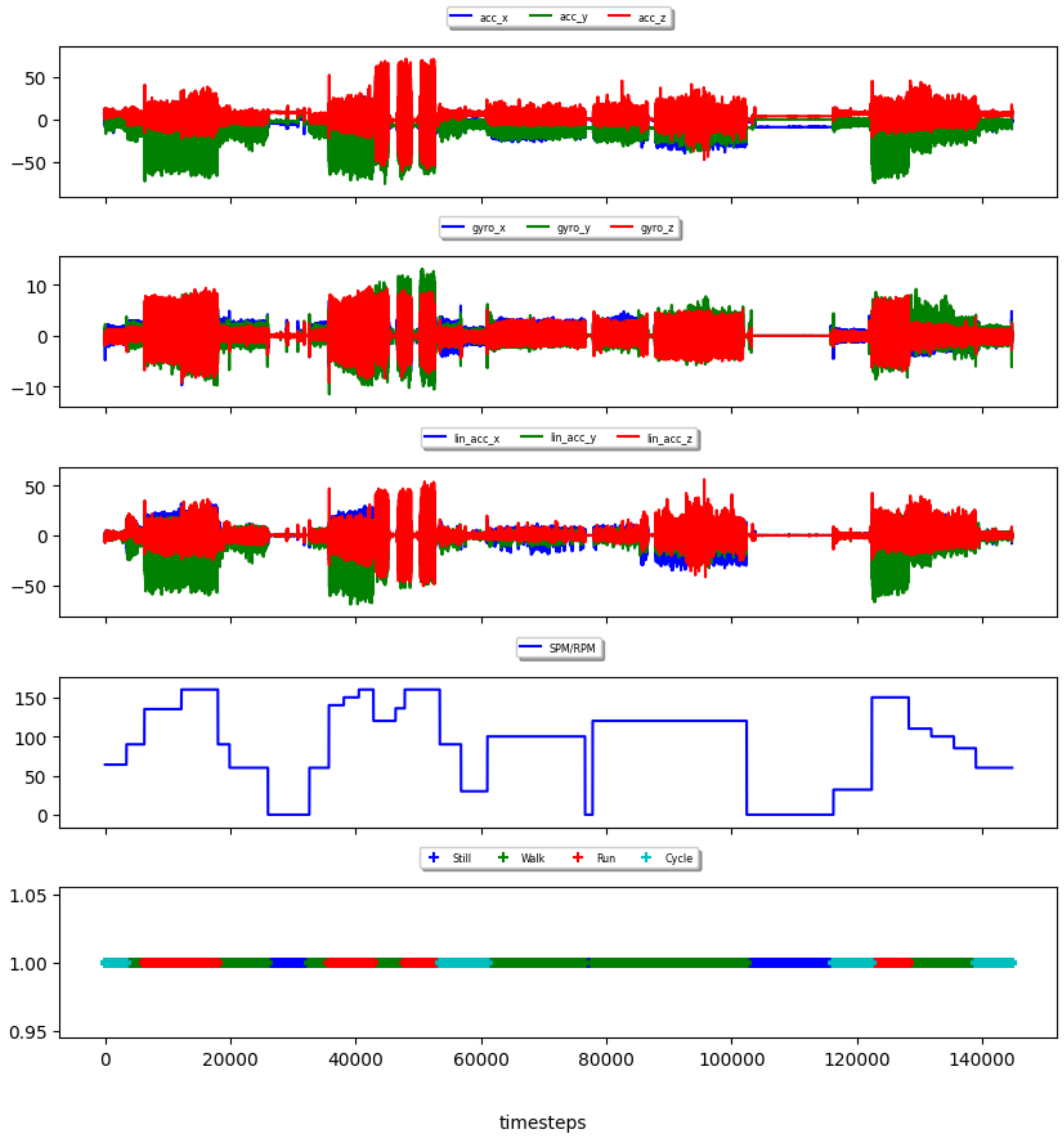
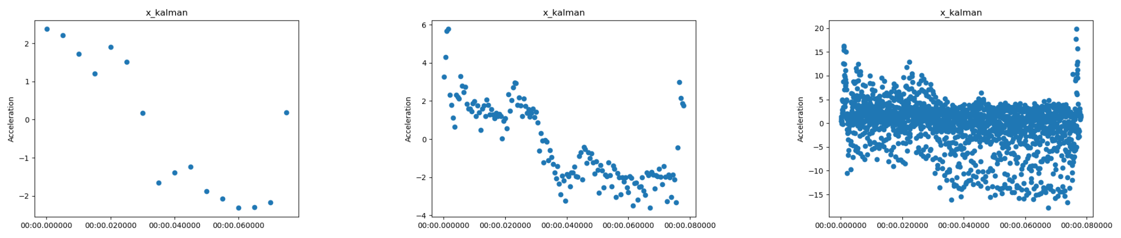


Fig. 2: Illustration of the dataset using 50ms timesteps

Table 2: Summary of Attribute Statistics

Attribute	Mean	Std	Min	Max
acc_x	-3.297792	6.776048	-53.098530	33.000000
acc_y	-7.994766	8.448560	-76.415688	19.051355
acc_z	1.965877	5.942238	-61.295143	71.680290
gyro_x	-0.000062	1.157910	-9.710000	7.852289
gyro_y	0.016088	1.465577	-11.500000	13.140241
gyro_z	0.018422	1.640554	-9.350000	9.720000
lin_acc_x	-0.165427	6.462054	-49.600000	32.100000
lin_acc_y	-1.180618	7.719546	-69.500000	22.200000
lin_acc_z	-0.638433	5.534859	-50.913715	56.441765

without an excessive number of datapoints within each vacillation. Figure 3 shows how the acceleration on the 'x' axis of movement appears at different granularities when averaging the timestamp values within a window. The ideal granularity is one for which a clear sinusoidal pattern appears.



(a) 5000ms (b) 500ms (c) 50ms

Fig. 3: Running: acceleration on the x-axis at granularities 5s, 500ms & 50ms

At 50ms the vacillation becomes more progressive, rather than jumping from one extreme to another, so it appears a strong place to start.

3 Oultier Detection and Missing values Imputation

During our investigation of outlier detection, we tried distribution-based methods, such as Chauvenet's criterion and mixture models. These methodologies usually thrive when the underlying data conforms to a normal distribution. To evaluate whether our dataset met this precondition, we conducted two well-established statistical tests for normality: the Shapiro-Wilk test and the D'Agostino's K-squared test. Both of these tests, unfortunately, dismissed the null hypothesis at a 0.05 significance level, suggesting that our dataset does not adhere to a normal distribution.

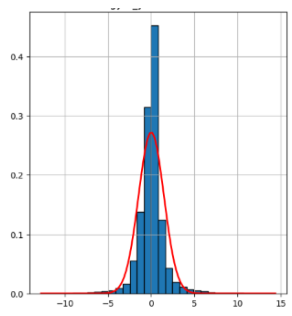
3.1 Inspecting Normality

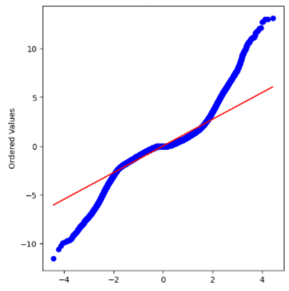
Both Chauvenet's criterion and the mixture models operate under the assumption of normality, meaning they perform best when the data follows a normal distribution. To verify whether our data adhered to this assumption, we conducted two widely used statistical tests for normality: the Shapiro-Wilk test and the D'Agostino's K^2 test. However, both these tests rejected the null hypothesis (that the data follows a normal distribution) at a significance level of 0.05, indicating that our data is not normally distributed.

This observation was further supported by the QQ (Quantile-Quantile) plot, a graphical tool used to assess if a dataset follows a particular theoretical distribution. In our case, the QQ plot demonstrated significant deviation from the line of equality (which represents the theoretical normal distribution), providing visual evidence that our data does not conform to a normal distribution. The illustaration of these evidence for Gyroscope's y values is provided in Figure 4b.

Moreover, an inspection of the histograms of our sensor readings did not reveal distinct distribution peaks, which would be expected if the data followed a mixture of normal distributions - the assumption underpinning the mixture model-based outlier detection.

Given these findings, we concluded that the Chauvenet's criterion and mixture models might not yield reliable results for outlier detection in our dataset.


(a) Distribution of Gyroscope y



(b) Gyroscope y QQ-Plot

Fig. 4: Illustration of the distribution diagnostics for Gyroscope's y values.

Considering the inability of distribution-based methods to accurately handle our dataset due to its deviation from normality, we turned our attention towards an alternative approach to manage outliers and impute missing values. In this context, we adopted the Kalman filter - a robust algorithmic solution that circumvents the need for normal distribution.

3.2 The Kalman filter

The Kalman filter is a recursive algorithm used to estimate the state of a dynamic system; in our case, the data from our phones, by effectively mitigating the noise within the data. The filter achieves this by initially predicting the future state based on past estimations. Then, when a new measurement is obtained, it updates this prediction by taking into account the new data. This continuous process of prediction and correction allows the Kalman filter to produce a statistically optimal estimate of the system's state. Essentially, the filter distinguishes between the state, which is not directly observed, and the measurements. The filtering is conducted through a set of transition equations that use measurements to describe the evolution of the state. We made the decision to use the Kalman filter since this approach deals with the outliers by looking at them in context. By leveraging this information, values with high volatility are discarded and replaced with more accurate predictions. The data gathered was expected to exhibit a critical frequency of oscillation corresponding to cadence. If true, the Kalman filter will help align extreme values better to this frequency. Figure 5 shows the Kalman filter's effect and that it regularizes extreme values.

4 Feature Engineering

Our data has a lot of information, which will make it difficult for some machine learning algorithms to process effectively. To create features that would be more useful to models like Deep Neural Networks with dense nodes, various Principal Component Analysis, Fourier Transformations, and sample distribution metrics were derived.

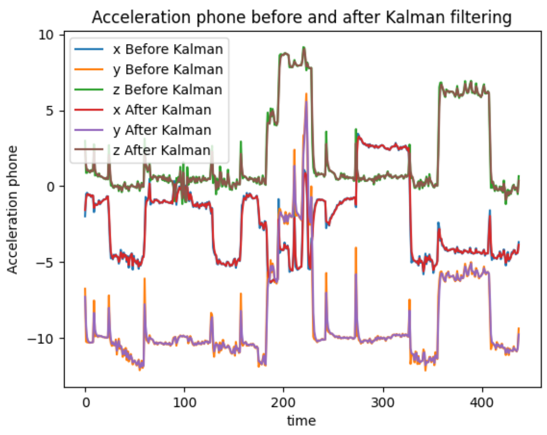


Fig. 5: The effect of the Kalman filter on a sample of data from Austin cycling.

4.1 Principal Component Analysis

We conducted PCA on our dataset, expecting to reduce the dimensionality. Linear Acceleration can be explained by the pull of gravity, regular acceleration, and gyroscopic orientation, so we expected to see dimensionality reduced by at least 3 with a high level of variance explained. Figure 2 shows how the number of principal components affects how much of the original variance is explained. We set our goal to explain over 0.9 of the variance and reached this with 6 components and 0.945 of variance explained. Figure 6 shows how the added components explain the original variance. 6 components is a fitting conclusion, because one can theoretically calculate linear acceleration from acceleration and gyroscope readings.

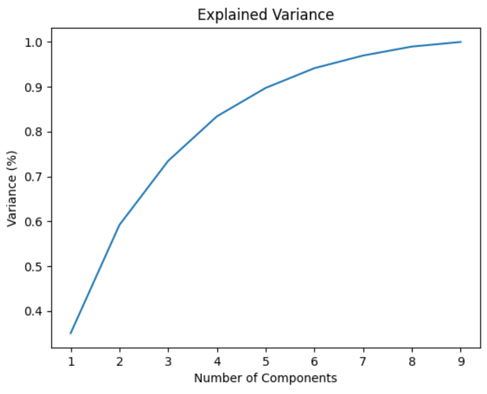
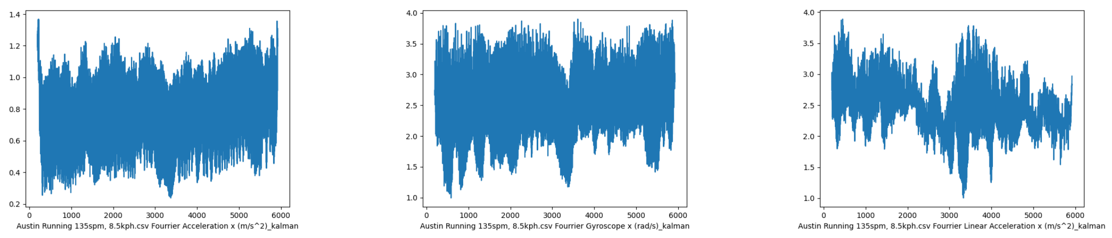


Fig. 6: This chart shows how each added component changes the explained variance from the original 9 variables. 6 components explain 0.945 of the variation in the original data.

4.2 Fourier Transformation

To determine the cadence of users, our algorithm looks at oscillation in the users movements that correspond to steps or revolutions. To this end, we extracted frequencies as features rather than noise. Our method follows the intuition that the strongest frequency the sensor detects is that of movement related to walking/biking/running, not any outside influence. We made Fourier transformations on our six principal components, looking at ten second windows in time. Our transformations took the strongest frequency in a window that corresponds to the sample size. Window sizes of 100, 200 and 400 timestamps (granularity 50ms) were completed to compare during the later machine learning steps. The window slides along the dataset with Kalman filtered PCA columns as inputs. On individual inspection, these frequencies are related to the cadence of the user, but not reliably. Figure 7 shows the difference between frequency and cadence for a running sample. Although the errors are evenly distributed, the mean absolute error is quite high at 45.



(a) Frequency of Component 1 (b) Frequency of Component 3 (c) Frequency of Component 5

Fig. 7: Frequencies during a user running at 135 SPM cadence

This means there are six total frequencies in each row of our dataset. When taking the average of these frequencies over an entire sample for an activity, the overall average corresponds loosely to the cadence in the label for that data. Figure 8 shows Fourier transformations across a full sample where the center line represents the average. The transformation changes over time, but a visual inspection shows that the average is very close to the true cadence label. This could be due to the labels for the data being imperfect, but accurate on average. We tried to step at a constant rate, but it was impossible to maintain absolute consistency. There could be other indications in the data, when the transformation goes high or low, which explains why we use other features to combine with the transformations for cadence prediction.

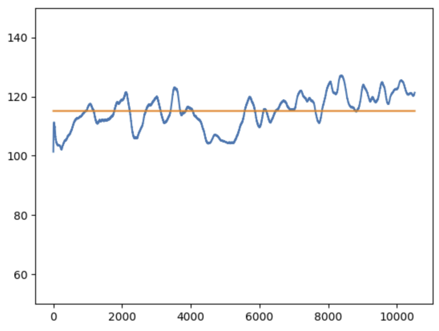


Fig. 8: This graphic shows the estimated frequency (Oscillations per min) at different windows from a sample that was specifically Austin running at a constant rate, and the average over all windows.

4.3 Datasets

A total of four datasets were created for the classification and regression tasks. For classification, all 23 activity samples were concatenated in a random ordering. This created a single csv file with a mix of activity types totaling about 2 hours.

For regression, only walking, running, and biking had required predictions. Each type used a different machine learning algorithm to make predictions. As such, all activity samples of just walking, just running, and just biking were concatenated into three separate datasets. This allowed samples for training the different prediction models to remain separate.

4.4 Train/Test Window Slices

Our models take a specific window in time into account when making predictions. This means that all the datapoints in that frame have the potential to be used for new features. The TCN and LSTM models used data from the PCA broken up into windows. However, for the Deep Neural Networks and Random Forest Classifier, we take the standard deviation, max and min values, mean, and max frequency for each column to create a sparse representation of the window. These sliced windows will represent inputs X_i for a target of activity type and cadence (if applies). The models we used look at each of these windows individually when making predictions. Figure 9 shows what a single frame would look like in terms of the original inputs. It is clear to the eye that some kind of frequency corresponding to pace is somewhere in this data. Window sizes of 100, 200, and 400 were used for comparison. The advantage of a larger window is generally less variation in frequency from one window to another. However, it is also less precise in moment to moment changes. An athlete may want to see minute changes in cadence and therefore favor a short window. For example, they may look at an assessment of their run using this tool to identify points in time where they shifted outside of their optimal cadence. This athlete could retain the insight and change their habits next time they run the same course. In theory, there is no reason that the user cannot make the window size selection themselves. Users going on a steady-pace activity may favor a large window, while track athletes may favor a very short window. We plan to make any window size that was proven to work available to users for predictions in the final product.

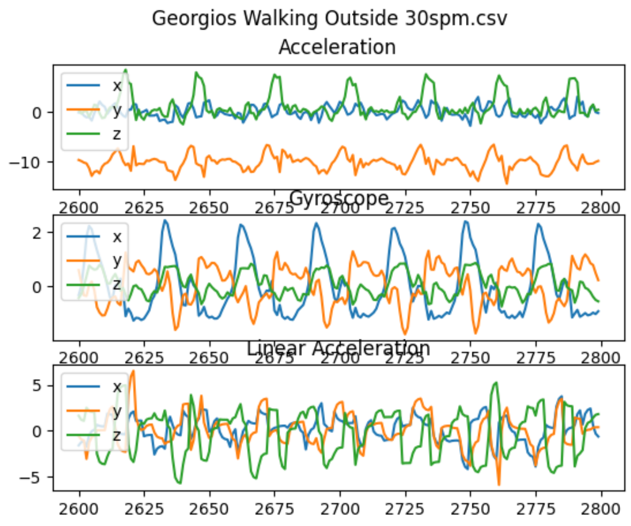


Fig. 9: This graphic shows all inputs graphed along 200 timestamps in a window.

5 Experimental Setup

Our models are designed to look at time windows individually. The task is to identify the type of activity, and then let the appropriate algorithm predict cadence if the activity is not sitting/rest. After creating features with window data distribution parameters, PCA, and the Fourier transformation, we trained different machine learning algorithms to see which can predict the labels accurately from our validation set. The sample windows for training activity classification and cadence estimation have the same format, but are sourced differently. Windows were produced using a 0.9 overlap from one window to the next. Therefore, with a windows of 100, 200, and 400 timestamps, we slide the window 10, 20, and 40 timestamps respectively, to get the next window. Training and testing split was taken at 0.8 to 0.2. The first nine frames from the test set are thrown out to prevent data leakage, because they overlap with the timestamps in the training set. The windows in each set are also shuffled after separation, so that they do not follow a strong correlation from one training input to the next.

5.1 Deep Neural Networks

Four separate Deep Neural Networks were implemented to classify activities and to predict cadence. Using Tensorflow, and drawing on experience with designing network architectures, performed grid search across a range of conventional hyperparameters. This includes learning rates [0.1, 0.05, 0.03, 0.02, 0.01, 0.005, 0.002, 0.001] with Adam optimizer, Dropout rates of [0.1, 0.2, 0.5], epochs of [1-10], node densities of [16, 32, 64, 128] per layer, and layer counts of [2, 3, 4]. The networks all take the same input, a sparse interpretation of the data within the prediction window. This makes an input shape of (5, 6), where the 5 rows correspond to max frequency, max value, min value, mean value, and standard deviation. The 6 columns correspond to the 6 principal components. The ideal hyperparameters were determined empirically by running a grid search over all combinations of hyperparameters. The best results were not the same every time, so this test was run multiple times for each of the four tasks. The grid search was completed until the same set of hyperparameters came up as ideal twice. For example, if the grid search was run three times, and the best performing Dropout rate for nodes was: 0.1, then 0.2, then 0.1 again, the dropout rate of 0.1 would be locked in after performing best a second time. The optimal hyperparameters for each of the four models are listed in the results section.

5.2 Random Forest Classifier

To serve as a benchmark for classification, a Random Forest Classifier (RFC) from the SciKit-Learn library was selected. This type of model requires little tuning or domain knowledge from the user before implementing, which makes it a handy baseline. This classifier used the same sparse inputs as the Deep Neural Networks, created 100 estimators and had a minimum of 1 sample per leaf. These are all default parameters in SKLearn.

5.3 Expanding the Training Data

Using a 0.9 overlap from one window to the next is good practice, but did not yield enough data to train TCN or LSTM models. Therefore, additional training data was created by taking the combined dataset and producing sample windows, while sliding the window just one row at a time. For LSTM and TCN regression models, all activities were combined before the windows were created. From all this data, a general solution across activity types was possible.

5.4 Temporal Convolutional Network

The TCN model is one well suited to the task of making a single prediction from a window of time, where many values are contained. This type of model is better oriented to handle a large input, such as a [200x6] matrix, because the dilation of the convolutional network is tailored to this window. In the context of this project, TCN is a more computationally intensive type of machine learning, but it sees much more of the variation in the original data. A TCN for classification and another for regression were both built into pipelines.

These networks require significant amounts of data to make predictions because the shape of inputs is so much larger. This large receptive field means capturing long temporal dependencies, which require many examples for the algorithm to learn because their relationship with the output is faint, yet relevant [3]. Due to significant training times, a grid search covering a huge variety of parameters was not possible. To minimize the chance of failing to find effective hyperparameters, a few quick comparisons were made between model parameters individually to get a rough set of hyperparameters. A conservative dropout rate of (0.2) was chosen to prevent overfitting while using 1 to 2 epochs. Also, a dimensionality of 64 was chosen for the TCN model, so there are 64 parallel units computing the input sequence. This was tested against 16 and 32 dimensional models, which performed poorly by comparison. With these parameters, a series or learning rates were tested at [0.03, 0.02, 0.01, 0.005, 0.002, 0.001, 0.0005].

5.5 Long Short Term Memory

The LSTM models followed a similar search design for parameters as the TCN, both had similarly long training times. Learning rates that led to a quick divergence were trimmed and the LSTM generally worked better with smaller learning rates than the TCN. However, the LSTM's performance overall was poor, so the slow learning rates may have just acted as a crutch, preventing the model from completely diverging.

6 Results

Looking at the results of all the models, we focus on accuracy for classification and mean absolute error for regression. Mean absolute error is a better fit than mean squared error in this application. MAE estimates of overall average cadence during an activity are equally affected by an error of 5 on a single point and an error of 1 on 5 separate points.

6.1 Random Forest Benchmark

The Random Forest Classifier was able to obtain 0.933 accuracy on the test set using the sparse inputs, this serves as a minimum to compare with other models. An important note is that this algorithm misclassified the vast majority of the resting/sitting samples. The confusion matrix for the RFC on the test set is shown in Figure 10

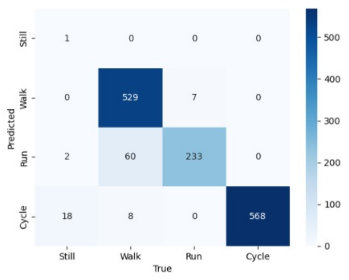


Fig. 10: Confusion Matrix of the RFC

6.2 Fourier Transform Benchmark

As a benchmark for the regression algorithms, we took an average of the max frequencies for the principal components in each sample window from each test set. These stay relatively close to the actual cadence except in running samples, as shown in Table 3. Thinking about the experimental design, during the running trials, the phone used for testing was likely getting bounced around. These tests were done on a treadmill while wearing athletic shorts. The excellent performance on the test set of the full dataset is an interesting find that gives a good challenge for other models to beat. Perhaps the samples that were most difficult to make predictions for did not end up in the last 20% of the dataset ordinally.

Table 3: Mean Absolute Error (MAE) for Different Activities

Activity	MAE
Run	96.817
Walk	21.215
Bike	20.013
Full Dataset	11.759

6.3 Deep Neural Networks

The dense layer neural networks performed well at their respective tasks, and patterns emerged in the architectures that performed best. While the classification algorithm was most accurate when using four dense layers, all the regression neural networks worked best with three layers. Considering max frequency correlates strongly cadence, the relationship between that frequency and the other distribution parameters is likely simpler. Whereas, the relationships between those window distribution parameters in the classification model are more important. Cadence ranges for biking and walking overlap significantly, which shows frequency alone is not nearly enough to make any sort of precise classification. The classification algorithm had an accuracy of 0.985, cutting total misclassifications to less than a quarter of the benchmark number.

Another interesting finding, the use of dropout had advantageous effects on the cycling and running regression algorithms, but not walking. This suggests that patters in predicting walking cadence are more generalizable, so the training data prepared the network well to predict from the test set, rather than overfitting. The walk, run, and bike cadence estimators had mean absolute errors of [6.189, 4.438, 8.96], respectively on their test sets.

The final architectures for the four dense neural networks are displayed in Table 4

Table 4: Hyperparameters for the DNNs

	Walk Regression	Run Regression	Bike Regression	Classification
Learning Rate	0.05	0.1	0.03	0.1
Epochs	10	8	10	1
Dropout	0.0	0.1	0.5	0.1

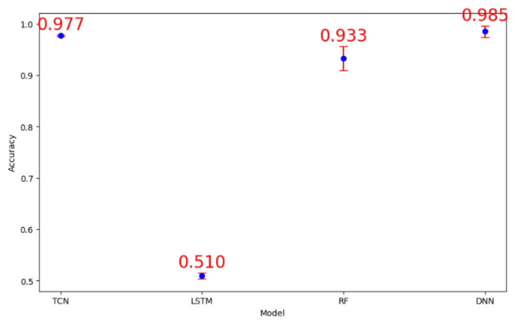
6.4 LSTM

The LSTM models tended to diverge during training. After trying a range learning rates, dropout, and LSTM dimensions, the best model could still only reach an test accuracy of about 0.51 on classification and a mean absolute error of 85.07 on regression. To train better functioning regression model, new training and test samples were made. They were samples taken from walking, running, and biking respectively, creating 3 separate train/test sets for regression. This time, the slide was set to 1 row to create enough data. With this separate data approach, the LSTM model was able to reach a mean absolute error of 50.45 for running, 50.39 for biking, and 52.43 for walking. These values are similar, in that they are very inaccurate.

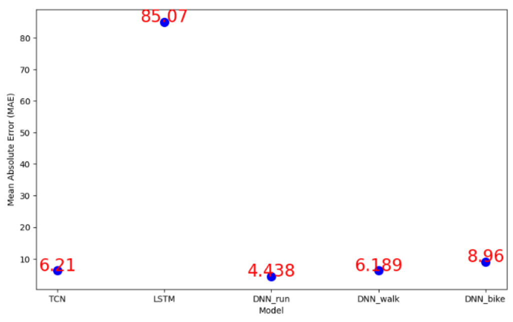
The architecture of the TCN is better suited to understand patterns in the data, and an LSTM can hold onto long range dependencies effectively. However, the mechanism of forgetting some previous information is counterproductive to understanding frequency, where each point is of more equal importance [5].

6.5 TCN

The TCN was ultimately the best model for regression and performed admirable in classification. It also took much more computing power than the other models that were able to converge. The classification model was able to predict activity type with an accuracy of 0.977 on the test set, giving it almost the third the number of misclassifications compared to the benchmark. The regression model scored a mean absolute error of 6.210 across all activities, beating any respective Fourier benchmark. The was both the best regression model and the most generalized, showing that more data points improve predictions, but at a large computational cost. The relative performances between models are shown in the confidence intervals of Figure 11. Notably, LSTM and TCN have very small confidence intervals because of much larger test sets.



(a) Performance Ranges, Classification Models



(b) Performance, Regression Models

Fig. 11: Performance Across All Models

7 Developing A Marketable Product

The results of our analysis prove this tool is an effective way of capturing the activity profile of a quantified self. However, the predictions themselves are information that need analysis. We constructed a simple tabulator, which organized the sample windows based on classification. The tabulator then estimated cadence by calling on the appropriate algorithm to make a prediction. After this process, the tabulator takes the number of windows in each activity to estimate the time spent total doing that activity for the day. Similarly, the overall cadence is estimated by taking the average over all estimates for the activity type. The tabulator was used to assess the entire labeled dataset, and the results are shown in Table 5.

There is room for much development from here. Graphics depicting the cadence of a user over time as a data visualization will come in handy for athletes who want to assess the consistency of their pace to optimize performance [7]. In a user application encapsulating our algorithms, a user

Table 5: Activity Summary

Activity	Duration (minutes)	Average Pace
Walking	45.9	72 steps per minute
Running	15.85	148 steps per minute
Biking	16.28	48 revolutions per minute

would also be able to slide the window size between the three proven windows. This is because there is less variance in the predictions in the longer window, making it useful for accurate readings during steady state activities. Conversely the higher variation in cadence estimation of the shorter window is useful for athletes who are pushing themselves and changing pace as they go.

Filtering the data and performing a Fourier transformation requires enough processing that it is very possible an Organization marketing this product would offer to do the processing on their own servers, opening the possibility for a subscription model. There is also the possibility of using a streamlined 'lite' version, which skips the kalman filter, making computation directly in the phone more feasible. The difference in accuracy of these algorithms without the kalman filter was not tested and is beyond the scope of this project, but given the overall similarity before and after as in Figure 5, the algorithms would likely perform reasonably well. This is especially true for the TCN model, because it does not take the max and min column values as a statistic as the DNN models do. The 'lite' version gives users an entry level option where they can decide if the app is really helping them reach their activity goals, before signing up to use the full model.

8 Discussion

In taking on this project, we investigated the connection between acceleration, gyroscope, and linear acceleration readings and user activity type, as well as cadence. We used Kalman filtering, Principal Component Analysis, and Fourer Transformations to extract important information from our data. Then we used Random Forest, Deep Neural Network, Temporal Convolutional Network, and Long Short Term Memory machine learning approaches to activity classification and cadence estimation. Our DNN classification model reached an accuracy of 0.985, our general TCN regression model got a mean absolute error of 6.21, and our DNN running regression model got a mean absolute error of 4.438.

If we had to take on this project again, we would first put more planning into the data collection process. This would help us to improve class imbalance, as correctly classifying the activity with the least data ('still/resting') proved difficult. During the other activities, we would also try to get more precise cadence estimations. The cycling was the most difficult type of activity to maintain a steady pace, and that was reflected in the final accuracy of our DNN regression models. To make this change, we would complete the cycling tests on less crowded roads, rather than around VU campus on a sunny afternoon.

During the data analysis, we would also look into more methods of outlier detection other than the Kalman filter. The Local Outlier Factor algorithm was not explored, but could have been an effective alternative to filtering. We would also try to assess the sensory readings as a time series and see if a ARIMA model can accurately predict the change over time. Then we could try to extract frequencies from the parameters of the model or even generate additional data to train models like TCN, which are data-ravenous.

Every aspect of this project could have been approached with more detail, given time. To create a real product of course takes far longer than a few weeks. In that time, we could gather a balanced dataset, inspect every aspect of our initial data, create more features, and ensure our models are using the best possible parameters. All of these actions would help our model to generalize to unseen users and cadences. However, with the time given, our final results were very accurate and certainly useful to any person who cares about their own health, athletic performance, and quantified self.

References

[1] *Data analysis in fitness: wellness through digital innovation.* Technogym - gym equipment and fitness solutions for home and business. Mar. 20, 2019. URL: `https://www.technogym.com/il/newsroom/fitness-future-data-analysis-wellness/`.

[2] George Grouios et al. *Accelerometers in Our Pocket: Does Smartphone Accelerometer Technology Provide Accurate Data?* Dec. 24, 2022. DOI: `10.3390/s23010192`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9824767/`.

[3] Luke Guerdan. *Diving Into Temporal Convolutional Networks.* URL: `https://lukeguerdan.com/blog/2019/intro-to-tcns/`.

[4] *How does a Gyroscope sensor work in your smartphone?* TechAhead. URL: `https://www.techaheadcorp.com/knowledge-center/how-gyroscope-sensor-work-in-smartphone/`.

[5] Bryan Tan. *Farewell RNNs, Welcome TCNs.* Medium. Sept. 2, 2020. URL: `https://towardsdatascience.com/farewell-rnns-welcome-tcns-dd76674707c8`.

[6] *The Coriolis Effect: Earth's Rotation and Its Effect on Weather.* URL: `https://education.nationalgeographic.org/resource/coriolis-effect` (visited on 06/22/2023).

[7] *Your Muscle Type and Cycling Fitness May Determine Your Ideal Cadence.* Bicycling. Section: Health & Nutrition. Dec. 14, 2022. URL: `https://www.bicycling.com/health-nutrition/a27454779/cycling-cadence-ideal/`.