Machine Learning Activity Classification

Author: Brendan Cotter

Due: December 9th, 2025

Problem Description And Dataset

The goal of this project was to classify human activity as jogging, walking, climbing upstairs, climbing downstairs, sitting, or standing. Accelerometer data in x, y, and z axes was collected from a subject's Android phone at a rate of 1 sample every 50 milliseconds. The data was collected in a controlled laboratory setting. In the raw laboratory dataset there were 1,098,207 samples and the class distribution was:

Walking - 424,400 - 38.6%

Jogging - 342,177 - 31.2%

Upstairs - 122,869 - 11.2%

Downstairs - 100,427 - 9.1%

Sitting - 59,939 - 5.5%

Standing - 48,395 - 4.4%

The dataset was then transformed by condensing 10 seconds of raw accelerometer data into a single sample. Each sample in the transformed dataset consisted of 200 accelerometer readings condensed into 46 values. A graph of a 10 second window of raw accelerometer readings is shown below.

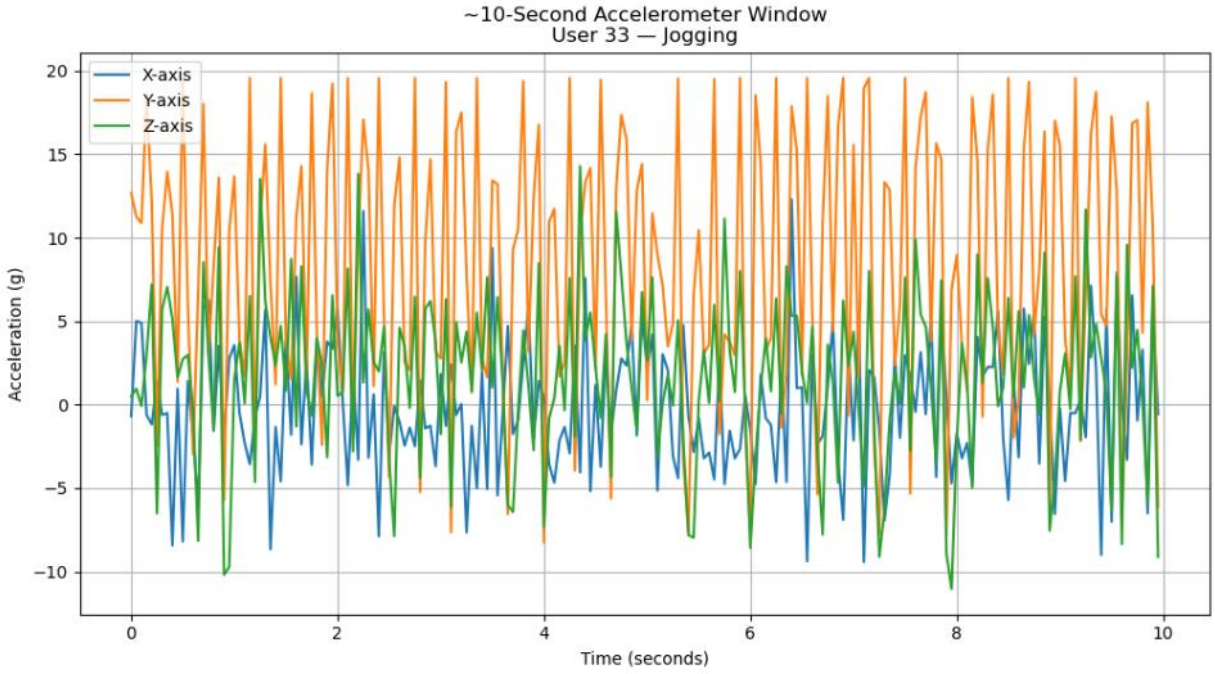


Figure 1: 10 Second Raw Accelerometer Window

Multi-layer perceptron neural networks and support vector machine (SVM) models could not be effectively trained using the raw dataset because the raw accelerometer readings lacked structured features suitable for supervised learning. These machine learning (ML) models also assume that each data point is independent of all other data points. However, in the raw dataset this was not the case because consecutive data points were highly correlated. Therefore, the raw dataset was transformed into the following statistical features so that ML models could be trained to classify activity.

The first 30 values in the transformed dataset were X0-X9, Y0-Y9, and Z0-Z9. These values were 1 second time-bins that contained the average acceleration in each direction for that second. These bins were utilized to capture the shape of the motion distribution over a 10 second window. For example, the bin X3 contained the average acceleration in the x-direction from 3 to 4 seconds. A histogram of the X3 bin is shown below. This feature clearly distinguishes sitting from walking upstairs because the 3 to 4 second average acceleration distributions for sitting and walking upstairs had almost no overlap.

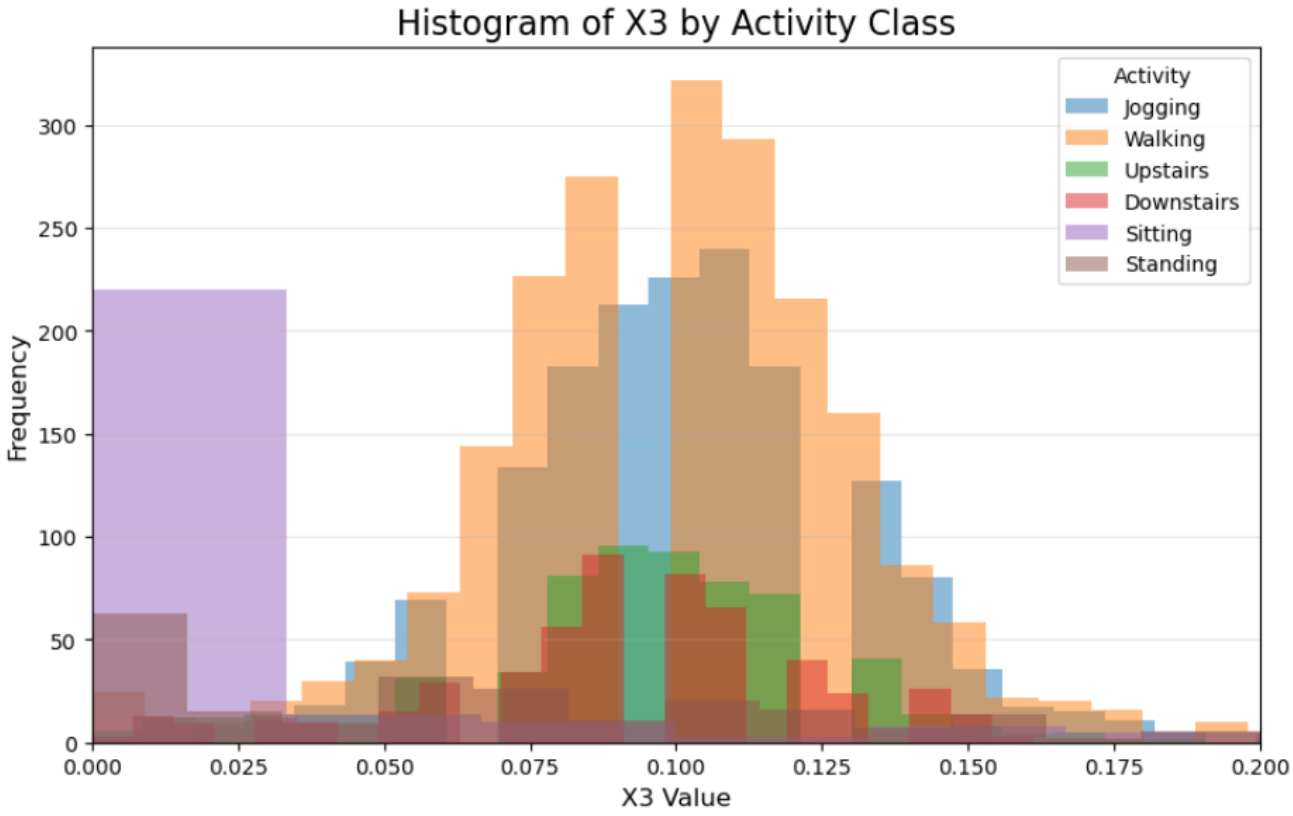


Figure 2: Histogram of X3 by Activity Class

X-PEAK, Y-PEAK, and Z-PEAK represented the dominant frequency of rhythmic motion detected in the accelerometer signal. The PEAK values were found by identifying the largest frequency amplitude in the sample. Then, all of the peak values within 10% of the dominant peak were identified. If fewer than three peaks were found, the threshold was iteratively lowered until at least three peaks were identified within a 10% range. The time between consecutive peaks was then summed and divided by the total number of peaks within a 10% range. X-PEAK, Y-PEAK, and Z-PEAK were utilized to capture rhythmic motion patterns. For example, jogging would exhibit strong periodic accelerations while sitting would exhibit little to no periodicity. The graph below indicated that the PEAK features were most useful for differentiating activities that involve motion from ones that do not involve motion. For example, the

magnitude of the peak frequency for sitting and standing was far less than the magnitude of the peak frequency for walking, running, climbing upstairs, and climbing downstairs.

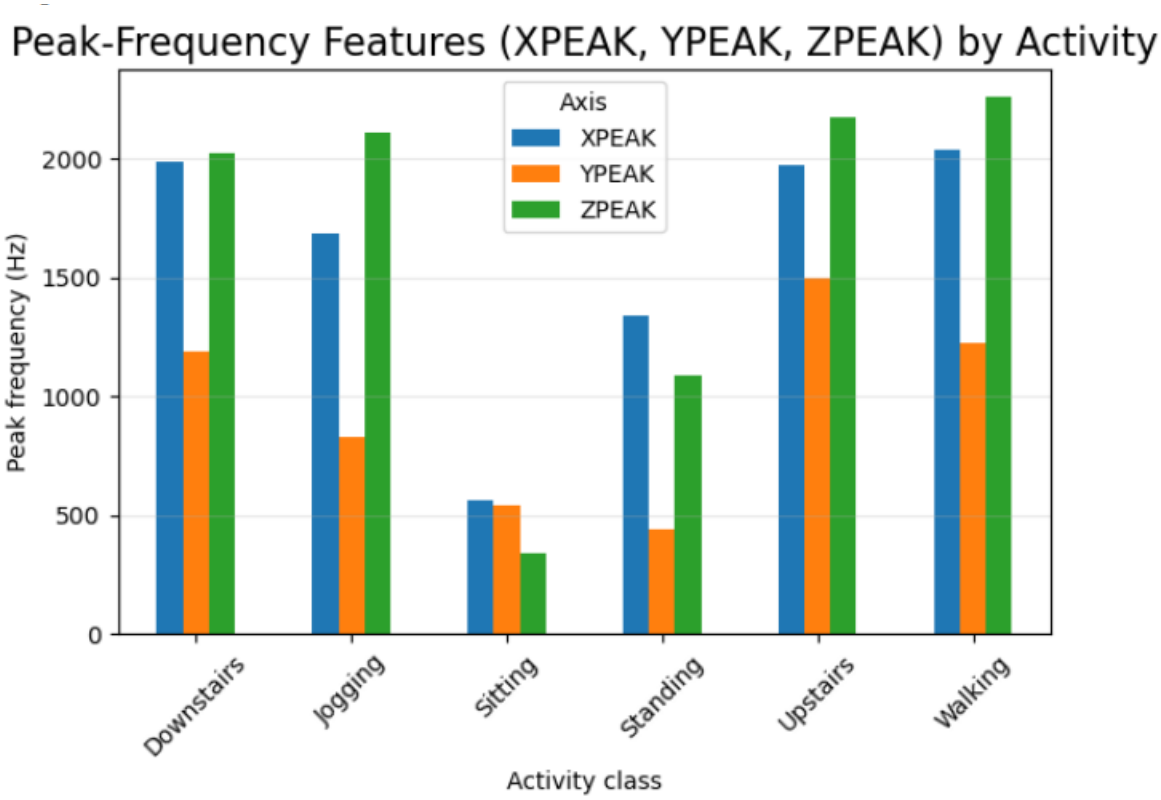


Figure 3: Peak-Frequency Feature Bar Chart (Activity Classification)

X-AVG, Y-AVG, and Z-AVG features represented the average accelerometer readings in the x, y, and z axes over a 10 second interval. X-STANDEV, Y-STANDEV, and Z-STANDEV measured the variability in acceleration along each axis. X-ABSOLDEV, Y-ABSOLDEV, and Z-ABSOLDEV measured the average absolute deviation from the mean along each axis. These features were utilized to determine the orientation of the subject's motion and as well as its intensity. The graph below illustrates how the x, y, and z standard deviations were useful for differentiating climbing downstairs from climbing upstairs. For example, the x, y, and z axis standard deviations for climbing downstairs were roughly 4 g, while jogging produced values closer to 8 g. These values were expressed in units of gravity (g) because in 2012 Android devices recorded acceleration normalized to gravitational units.

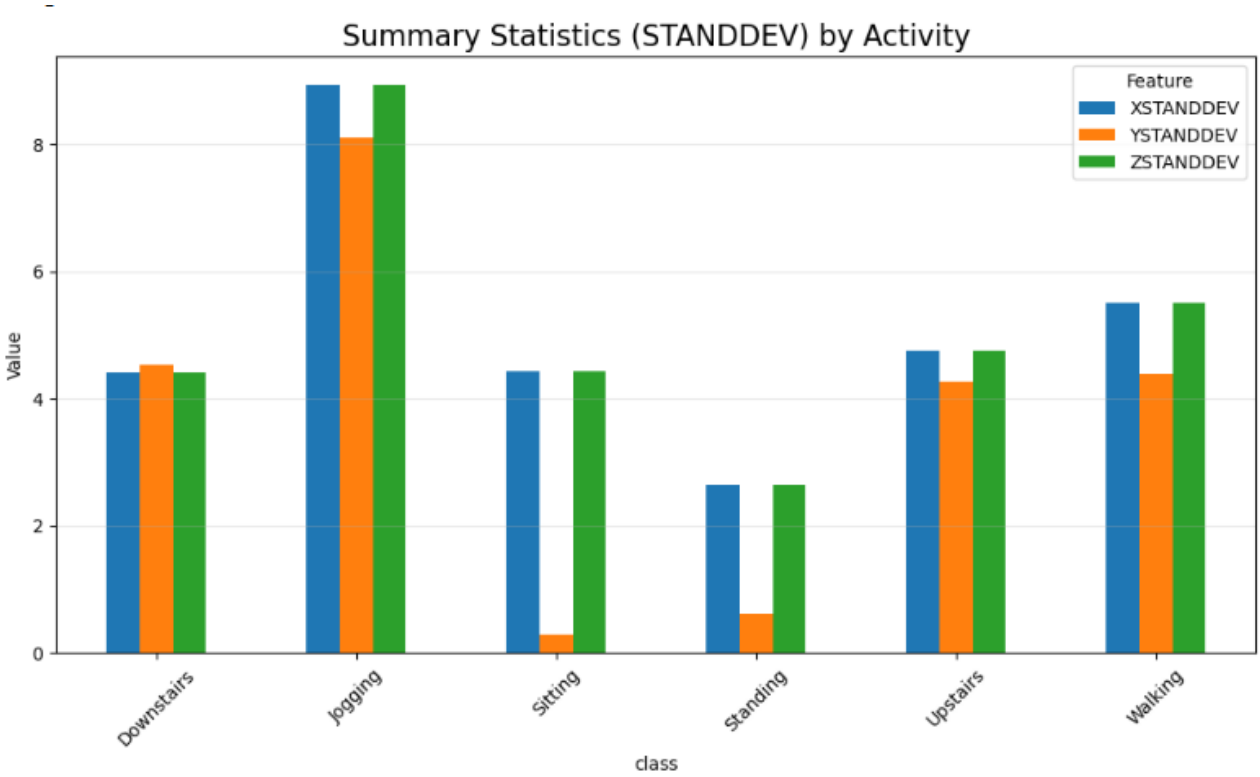


Figure 4: STANDDEV Feature Bar Chart (Activity Classification)

The RESULTANT measured the magnitude of total acceleration and was found via this equation:

$$R = \sqrt{(X - AVG)^2 + (Y - AVG)^2 + (Z - AVG)^2}$$

The remaining features were user, UNIQUE_ID, and class. The user feature was utilized to pair each individual participant with the data that was produced. Each participant performed various activities (walking, jogging, etc.). Therefore, the UNIQUE_ID was utilized to distinguish a participant's walking samples from their jogging samples. The class feature was utilized to identify the activity the user was partaking in at the time the data was collected. In the transformed laboratory dataset there were 5,424 samples and the class distribution was:

Walking - 2,082 - 38.4%

Jogging - 1,626 - 30.0%

Upstairs - 633 - 11.7%

Downstairs - 529 - 9.8%

Sitting - 307 - 5.7%

Standing - 247 - 4.6%

The data set described above was created by Dr. Gary Weiss's WISDM Lab at Fordham University (WISDM Lab). The title of the data set was "Activity Prediction" and the last time it was updated was December 2nd, 2012 (Activity Prediction Data Set). This dataset was collected in a controlled laboratory setting. As a result, a key limitation of training motion classification ML models on this data was that they did not generalize well to data generated in a real-world setting. Another limitation of the "Activity Prediction" dataset was the uneven class distribution. There were far more samples for walking and jogging compared to sitting and standing. Therefore, the best performing ML models created in this project would likely perform worse on sitting and standing data versus walking and jogging data. This would be due to the fact that the ML models had less sitting and standing data to be trained on.

The WISDM Lab also created a data set titled "Actitracker" which contained real world data that was both labelled and unlabeled. This dataset was also utilized for this project (Actitracker Data Set). In the real world transformed dataset the labelled data was separated from the unlabeled data. Thus, the accuracy of the best performing ML models was also evaluated on real word data. This test was valuable for assessing the real-world applicability of the best performing ML models created in this project. In the raw real world dataset there were 2,980,765 samples and the class distribution was:

Walking – 1,255,923 – 42.1%

Jogging – 438,871 – 14.7%

Stairs – 57,425 – 1.9%

Sitting – 663,706 – 22.3%

Standing – 275,967 – 9.7%

LyingDown – 275,967 – 9.3%

In the transformed real world dataset there were 5,424 samples and the class distribution was:

Walking – 2,185 – 40.2%

Jogging – 130 – 2.4%

Stairs – 251 – 4.6%

Sitting – 1,410 – 25.9%

Standing – 840 – 15.5%

LyingDown – 619 – 11.4%

One of the limitations of the real world dataset was that it had different activity classes than the laboratory dataset. The real world dataset combined the activities of climbing upstairs and downstairs into climbing stairs. It also included the activity LyingDown which was not present in the laboratory dataset. Therefore, these classes had to be removed in order to evaluate the accuracy of the ML models trained on the laboratory data.

The applications for motion classification ML models are vast. For example, they could be used by fitness trackers such as Fitbit or Garmin watches. Users of these devices could then determine the amount of time they spent jogging, walking, climbing upstairs, climbing downstairs, sitting, and standing each day. Motion classification ML models could also be utilized for healthcare applications. A patient recovering from ACL surgery could use this system to track how much time they spend performing different daily activities. This would provide an objective measure of their mobility and progress throughout rehabilitation. Motion classification ML models could also be utilized by athletes in order to optimize their workouts to achieve the best improvement in their physical performance. These ML models could also be utilized in robotics. Motion classifier ML models could be utilized to determine if a robot was functioning correctly.

Motivation For Using Machine Learning

Using machine learning for motion classification is appropriate because prediction errors are unlikely to cause harm to users. For example, a fit bit user would not be physically damaged if a cool down walk after a run was misclassified as jogging. The user would just be confused when they saw that their fit bit claimed they ran for 25 minutes when they know they went on a 20 minute run.

Human motion is also complex and variable. For example, a short person and tall person would have very different gaits and jogging rhythms. Even when people are the same height there are variables that may cause them to have different gaits and jogging rhythms. For example, a healthy 5’8” male would not walk or jog the same as a 5’8” male recovery from ACL reconstruction. Machine learning models could address this issue by training on data specific to each user. Although this approach was not utilized in this project, it could be easily implemented by separating the dataset based on each user.

The two different ML models utilized in this project were support vector machines (SVM) and multi-layer perceptron (MLP) neural networks (NN). The three different kernels that were utilized for the SVM models were a linear kernel, a polynomial kernel, and a radial basis function (RBF) kernel. The four different activation functions utilized for the MLP NN models were the ReLU, Leaky ReLU, Tanh, and Sigmoid activation functions.

Support Vector Machine (SVM)

Support Vector Machines are machine learning models that find a decision boundary that separates classes. An SVM utilized in tandem with a radial basis function (RBF) kernel makes it possible to map features into a higher dimensional space where they are linearly separable. SVMs inherently include regularization which balances margin width and classification accuracy. The optimal margin classifier determines the optimal hyperplane that best separates the classes of data in high dimensions.

Linear Kernel

The linear kernel attempted to find a flat hyper plane decision boundary that could separate the activity classes. The linear kernel could not find nonlinear interactions between features that could separate the activity classes. A general formula for the linear kernel is shown below where x represents one data point with 43 features and z represents another data point with 43 features:

$$K(x_1, x_2) = x_1 * x_2$$

$$x_1 = \text{One Datapoint} \quad x_2 = \text{A Second Different Datapoint}$$

Polynomial Kernel

The polynomial kernel attempted to find a curved hyper plane decision boundary that could separate the activity classes. However, the complexity of these curved decision boundaries were constrained by the polynomial degree that was chosen. The polynomial kernel could find nonlinear interactions between features. However, the complexity of the nonlinear feature interactions were also limited by the polynomial degree that was chosen for the kernel. A general formula for the polynomial kernel is shown below:

$$K(x_1, x_2) = (x_1 * x_2 + c)^d$$

$$c = \text{Bias Term} \quad d = \textit{Polynomal Degree}$$

RBF Kernel

The RBF kernel also attempted to find a curved hyper plane decision boundary that could separate the activity classes. The RBF kernel made it possible to map the features to a higher dimensional space where they are linearly separable and could utilize polynomials to the infinite degree. Therefore, the RBF kernel could find complex nonlinear interactions between features that could separate the activity classes. A general formula for the polynomial kernel is shown below:

$$K(x_1, x_2) = e^{(\gamma||x_1 * x_2||^2)}$$

$$\gamma = \text{Kernel Width}$$

Multi-Layer Perceptron Neural Networks (MLP NN)

MLP neural nets are supervised learning models that classify data by finding nonlinear transformations between the input features. An MLP consists of multiple layers of interconnected neurons that each apply a nonlinear activation function and weighted sum to the neuron's input. Activation functions are utilized to determine nonlinear interactions between features. Regularization was included in these models in the form of L2 penalties. L2 penalties prevented the MLP neural nets from memorizing noise in the data and allowed them to better extract generalizable patterns in the data.

ReLU

The ReLU function outputs zero for negative inputs and grows linearly for positive inputs. This activation function is useful for training neural nets efficiently and performs best on datasets that are sparse with a high variance. It is also useful for addressing the issue of vanishing gradients which can occur in deep neural networks.

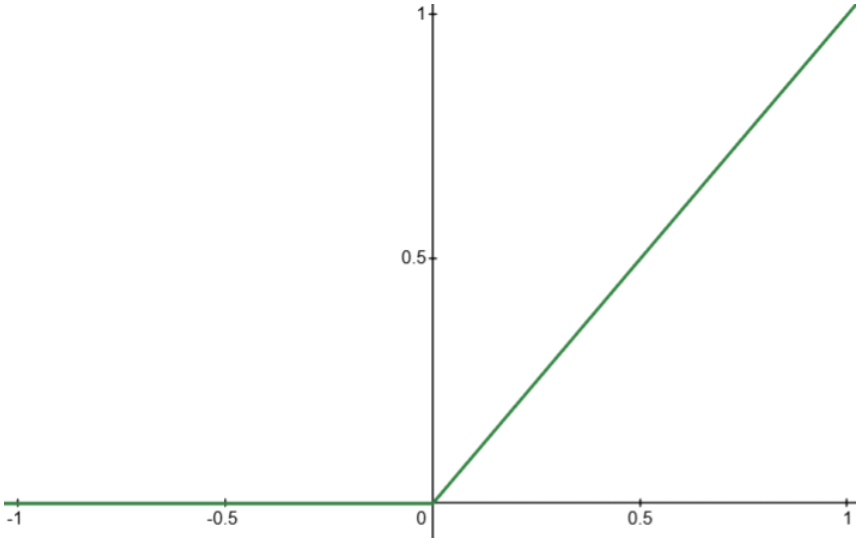


Figure 5: ReLU Activation Function

Leaky ReLU

The Leaky ReLU function grows linearly for positive inputs and with a small, shallow slope for negative inputs. This activation function is similar to ReLU. However, it is slightly more flexible because it does not assign negative inputs a gradient of 0. This prevents neurons from becoming inactive during training.

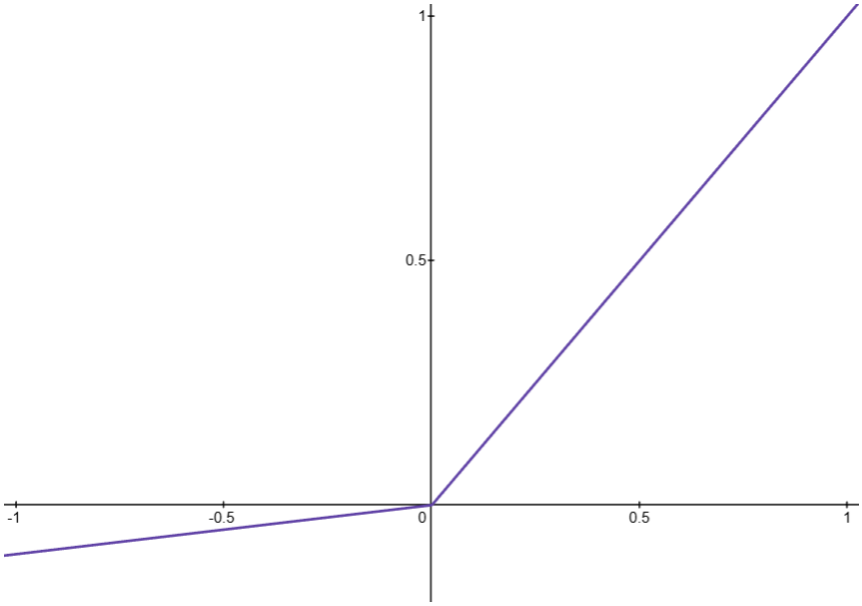


Figure 6: Leaky ReLU Activation Function

Sigmoid

The Sigmoid function squeezes its output range to values between 0 and 1. This activation function performs best on datasets with smooth, continuous feature relationships.

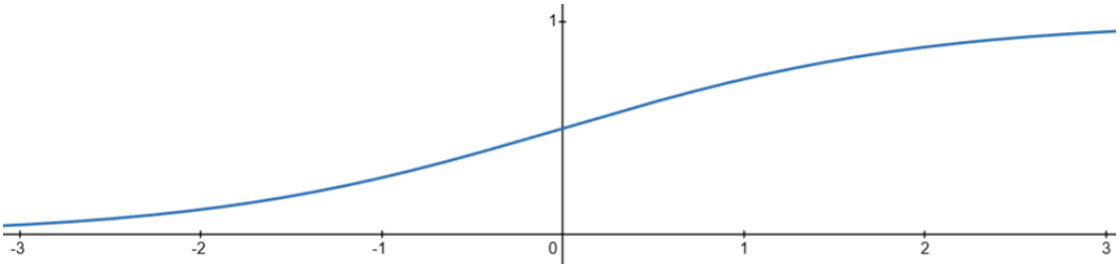


Figure 7: Sigmoid Activation Function

Tanh

The Tanh function is similar to the sigmoid activation function. This activation function also performs best on datasets with smooth, continuous feature relationships. However, unlike the Sigmoid function it is centered at zero and outputs values between -1 and 1.

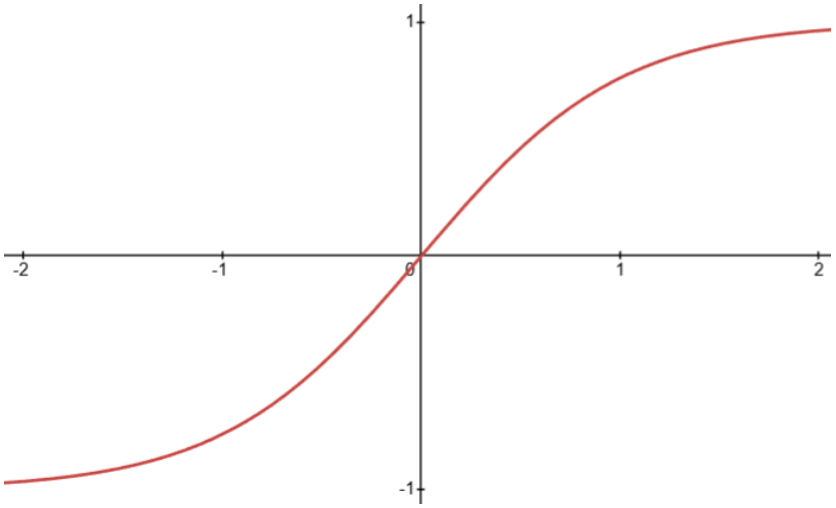


Figure 8: Tanh Activation Function

Training Process

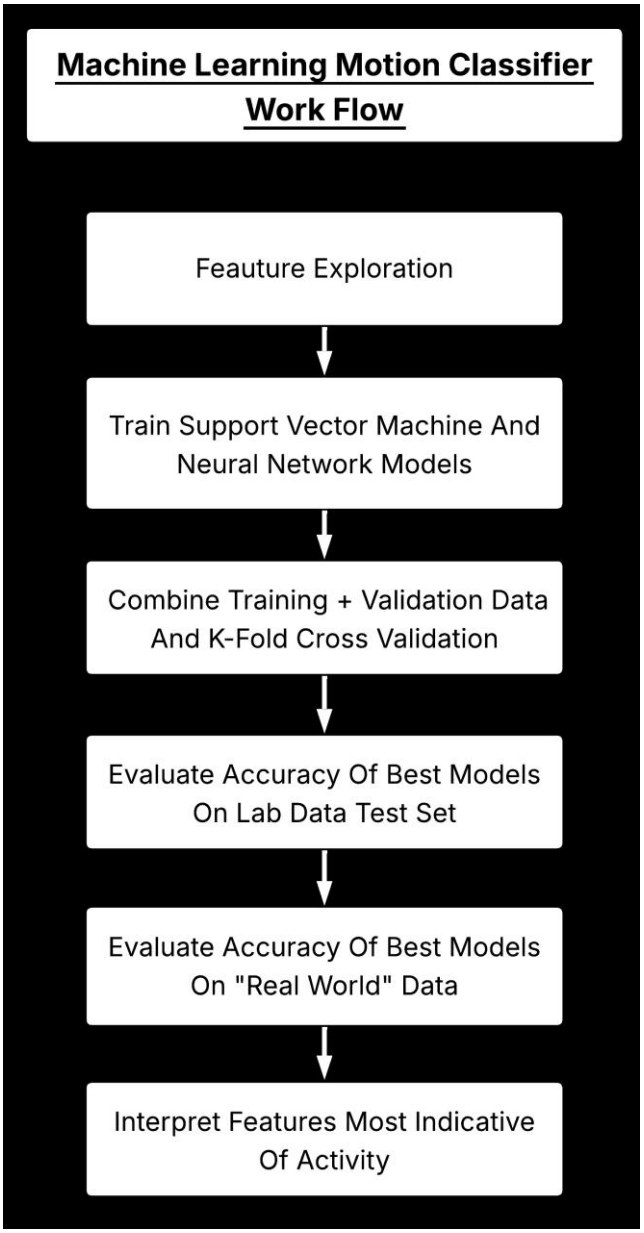


Figure 9: Machine Learning Motion Classifier Workflow

To begin this project the "Activity Prediction" zip file was downloaded from the WISDM Lab's website. The transformed data set was then extracted and loaded into a Jupyter notebook. Histograms of X0-X9, Y0-Y9, and Z0-Z9 were then generated. Barcharts of X-PEAK, Y-PEAK, Z-PEAK, X-AVG, Y-AVG, Z-AVG, X-STANDEV, Y-STANDEV, Z-STANDEV, X-ABSOLDEV, Y-ABSOLDEV, and Z-ABSOLDEV were also created. These graphs were created in order to qualitatively determine which features may be indicative of activity.

The transformed data was then split into a training, validation, and test set with a standard 60-20-20 split. The data was also standardized in order to prevent features with large magnitudes from dominating the machine learning process. The formula utilized to scale the features is shown here:

$$x_j' = \frac{x_j - \mu_j}{\sigma_j}$$

SVM models with linear, polynomial, and RBF kernels were then trained and validated on the data splits. The SVM models were trained over a hyperparameter grid in order to determine which kernel performed the best.

MLP neural net models with ReLU, Leaky ReLU, Tanh, and Sigmoid activation functions were then trained and validated on the data splits. The MLP neural net models were trained over a hyperparameter grid in order to determine which activation function performed the best.

The training and validation datasets were then combined and k-fold cross validation was performed on an SVM RBF model, a MLP NN-Tanh model, and a MLP NN-Sigmoid model. K-fold cross validation was utilized in order to determine the optimal hyperparameters for these models. These three models were then trained on the combined training and validation dataset after the optimal hyperparameters for each model were determined. The models were then evauluated on the laboratory test dataset in order to determine their accuracy.

Next, the "Actitracker" zip file was downloaded from the WISDM Lab's website. The transformed data set was then extracted and loaded into a Jupyter notebook. The "LyingDown" and "Stairs" activity classes were then removed from the dataset because the ML models were not trained on data that included these activities. The transformed dataset was then scaled using the same scaler that the transformed laboratory dataset used. The best three ML models were then evaluated on the real world data in order to determine their accuracy.

Permutation importance was then applied using the SVM model with an RBF kernel to identify which features were most indicative of activity. Permutation importance sequentially breaks each feature's relationship with activity classification and then evaluates the resulting drop in model accuracy. The features that produced the largest decrease in accuracy were considered to be the most important for activity classification. The top three most important features were then plotted against one another to assess whether the activity classes were linearly separable. Additionally, density plots were generated to visualize class distributions and highlight which activities would require a nonlinear decision boundary to distinguish effectively.

Results And Analysis

The histograms of X0-X9, Y0-Y9, and Z0-Z9 suggested that these features were useful for differentiating sitting and standing from climbing upstairs, climbing downstairs, walking and jogging (Appendix A). The bar charts for X-AVG, Y-AVG, Z-AVG, X-STANDEV, Y-STANDEV, Z-STANDEV, X-ABSOLDEV, Y-ABSOLDEV, and Z-ABSOLDEV suggested that these features were useful for differentiating all 6 activities (Appendix B). The bar charts for X-PEAK, Y-PEAK, and Z-PEAK suggested that these features were useful for differentiating sitting and standing from climbing upstairs, climbing downstairs, walking and jogging (Appendix C).

Listed in table 1 below are the hyperparameters and validation accuracies for the SVM models trained with linear, polynomial, and RBF kernels. The best performing SVM linear kernel model had an accuracy of 77.86% and had a regularization strength of 10. The best performing SVM polynomial kernel model had an accuracy of 83.67% and had a regularization strength of 10. The best performing SVM RBF kernel model had an accuracy of 84.13% and had a regularization strength of 100.

Best Performing Support Vector Machines			
Kernel	**λ (Regularization Strength)**	**Kernel Width**	**Validation Accuracy (%)**
Linear	10	N/A	77.86
Polynomial	10	0.01	83.67
RBF	100	0.01	84.13

Table 1: Best Performing SVM Models

The RBF kernel outperformed both the linear and polynomial kernels on the validation set. This was likely because the RBF kernel was well suited for capturing the nonlinear and locally clustered patterns present in the data. Unlike the polynomial kernel, which was limited to the 4th-degree, the RBF kernel implicitly modeled interactions of all polynomial degrees. This allowed the RBF kernel to represent far more complex nonlinear relationships. In contrast, the linear kernel could not capture any nonlinear interactions at all.

Listed in table 2 below are the hyperparameters and validation accuracies for the MLP neural net models trained with ReLU, Leaky ReLU, Tanh, and Sigmoid activation functions. The best performing MLP NN-ReLU model had an accuracy of 84.69% and had a regularization strength of 0.0001. The best performing MLP NN-Leaky ReLU model had an accuracy of 81.37% and had a regularization strength of 0. The best performing MLP NN-Tanh model had an accuracy of 85.24% and had a regularization strength of 0.001. The best performing MLP NN-Sigmoid model had an accuracy of 85.24% and had a regularization strength of 0.01.

Best Performing Multi-Layer Perceptron Neural Networks				
Activation Function	**# Hidden Layers**	**λ (Regularization Strength)**	**α (Learning Rate)**	**Validation Accuracy (%)**
ReLU	2 [128, 64]	0.0001	0.01	84.69
Leaky ReLU	2 [256, 128]	0	0.01	81.37
Tanh	3 [256, 128, 64]	0.001	0.01	85.24
Sigmoid	3 [128, 64, 32]	0.01	0.01	85.24

Table 2: Best Performing MLP NN Models

The MLP neural networks with Tanh and Sigmoid activation functions performed the best on the validation set. They outperformed the neural nets that used ReLU and Leaky ReLU activation functions. This likely occurred because

the input features were smooth and continuous accelerometer statistics. Tanh and Sigmoid activation functions worked well with this type of data because they produced gradual changes in output when the input changed slightly. ReLU activation functions on the other hand output zero for many values near zero and did not respond as precisely to small input changes. Also, the neural nets in this project were only three layers deep. Therefore, vanishing gradients were not a major issue. Thus, the Tanh and Sigmoid activation functions were effective for training the neural network models.

The SVM RBF model, MLP NN-Tanh, and MLP NN-Sigmoid models were then selected to move on to k-fold cross validation. The training and validation datasets were combined for k-fold cross validation in order to maximize the data available for learning the optimal hyperparameters of these models. After the optimal hyperparameters had been determined the models were trained on the combined training and validation dataset. These three models were then evaluated on both the laboratory test data and real world data.

Final Support Vector Machine Model				
Kernel	λ (Regularization Strength)	Kernel Width	Lab Data Accuracy (%)	Real World Data Accuracy (%)
RBF	100	0.01	84.23	30.93

Table 3: Final SVM Model

Final Multi-Layer Neural Networks					
Activation Function	# Hidden Layers	λ (Regularization Strength)	α (Learning Rate)	Lab Data Accuracy (%)	Real World Data Accuracy (%)
Tanh	5 [256, 256, 128, 64, 32]	0.01	0.01	86.35	27.25
Sigmoid	3 [256, 128, 64]	0.001	0.01	86.35	27.67

Table 4: Final MLP NN Models

The final SVM RBF model reached an accuracy of 84.23% on the laboratory test set, but its performance decreased to 30.93% on the real world dataset. The final MLP NN-Tanh model reached an accuracy of 86.35% on the laboratory test set, but its performance decreased to 27.25% on the real world dataset. The final MLP NN-Sigmoid model reached an accuracy of 86.35% on the laboratory test set, but its performance decreased to 27.67% on the real world dataset.

Final Machine Learning Models		
Model	Lab Data Accuracy (%)	Real World Data Accuracy (%)
SVM-RBF Kernel	84.23	30.93
MLP NN-Tanh	86.35	27.25
MLP NN-Sigmoid	86.35	27.67

Table 5: Final Models Accuracy Summary

The neural network models slightly outperformed the SVM model on the laboratory test set. This was likely due to the fact that the neural network models were more flexible. The layered structure of the neural network models allowed them to better capture nonlinear interactions between features. However, the specific feature interactions learned by the neural network models remained unknown due to the fact that these models were somewhat of a black box. In contrast, the SVM model relied on a fixed kernel shape in order to determine nonlinear feature interactions. This limited the ability of the SVM model to capture complex interactions between features. Surprisingly, the SVM

model performed the best on the real world data. However, this was just relative to the neural nets because none of the models were able to achieve an accuracy greater than 31%. The main takeaway from this test was that the activity classification models trained on controlled laboratory data did not generalize well to real world conditions.

Permutation importance with the optimal hyperparameters for a SVM RBF model was then performed. This process determined that the top features most indicative of activity were YABSOLDEV (vertical acceleration fluctuation), YSTEANDEV (intensity and rhythmicity of motion), ZABSOLDEV (tilt acceleration fluctuation), YAVG (vertical acceleration intensity), and RESULTANT (magnitude of total acceleration). YABSOLDEV was then plotted against YSTEANDEV and ZABSOLDEV. Density plots were also created for these graphs. These graphs showed that some of the activity classes were linearly separable, like jogging (blue) and standing (brown). However, the distributions for activities like climbing upstairs (green) and climbing downstairs (red) were nearly on top of each other. Therefore, a nonlinear hyperplane would be needed in order to separate these classes.

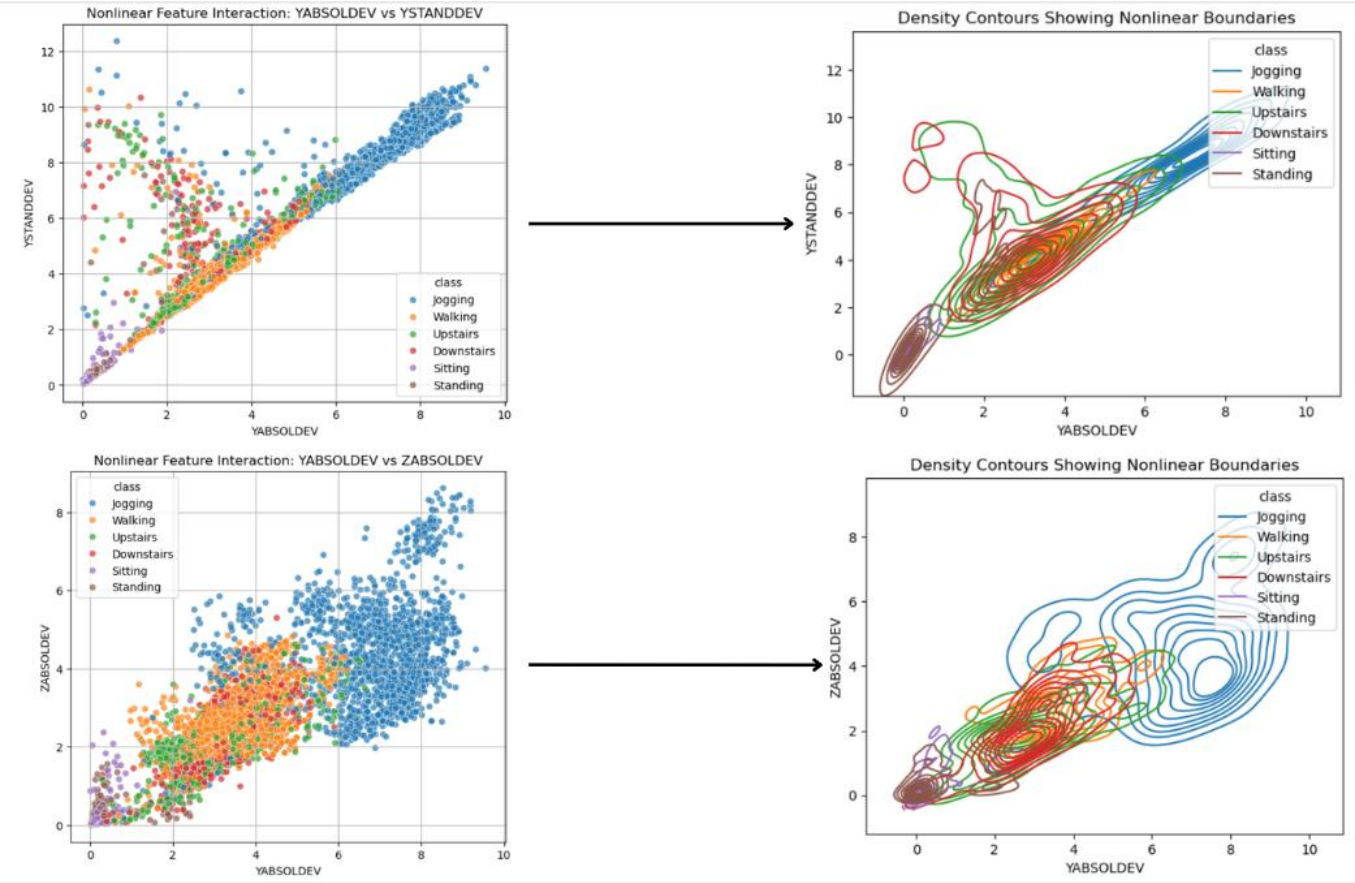


Figure 10: Feature Interaction Graphs

Conclusion

Multi-layer perceptron neural network models were able to most accurately classify motion. This was likely due to the fact that they are slightly more complex than SVM models and can capture layered, nonlinear interactions between features. Also, the motion classifying machine learning models did not perform well when tested on real world data. This was likely due to the fact that the data collected in a laboratory setting did not accurately mimic real world conditions. The final takeaway from this project was that YABSOLDEV (vertical acceleration fluctuation) was the feature that was most indicative of activity. This made sense because activities such as jogging, walking, climbing upstairs, and climbing downstairs would all have varying levels of vertical acceleration.

One of the limitations in this project was the uneven class distribution in the transformed dataset. The activities of standing and sitting accounted for only about 10% of the transformed dataset, despite representing 33% of the possible activity classes. As a result, the machine learning models likely struggled to classify these activities accurately due to insufficient training examples. A potential solution would be to use a generative model to create synthetic data for the underrepresented classes. However, generative models typically require large amounts of training data to produce high quality synthetic data. Therefore, any generated data would need to be validated carefully before being added to the dataset. One approach would be to train ML models exclusively on the synthetic samples and then evaluate their performance on real data. If the models trained on synthetic data performed well on real examples then the synthetic data would be considered valid.

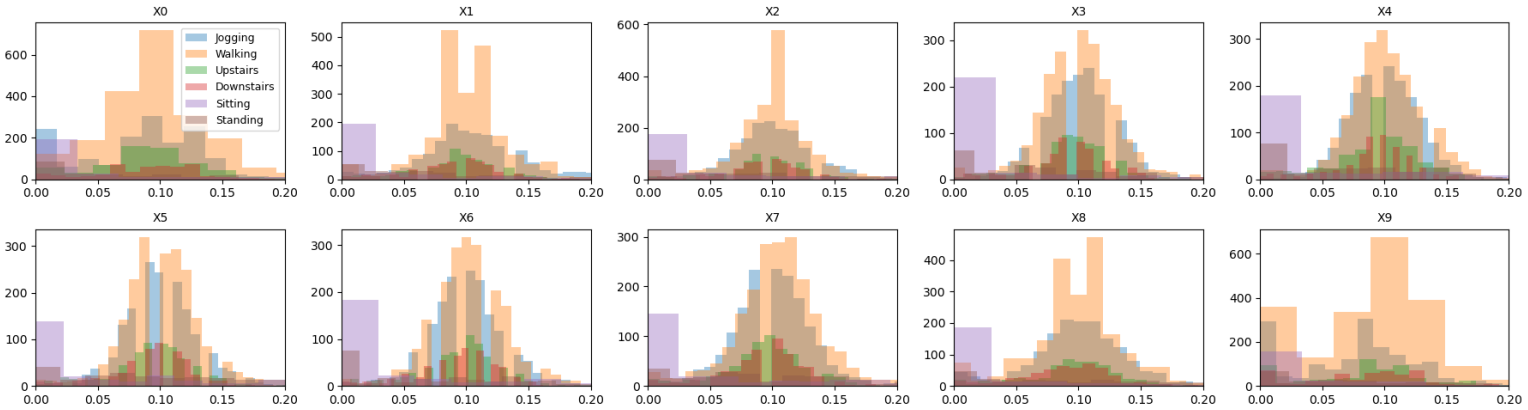
Citations

1. Jennifer R. Kwapisz, Gary M. Weiss and Samuel A. Moore (2010). Activity Recognition using Cell Phone Accelerometers, *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10)*, Washington DC.
2. Jeffrey W. Lockhart, Gary M. Weiss, Jack C. Xue, Shaun T. Gallagher, Andrew B. Grosner, and Tony T. Pulickal (2011). "Design Considerations for the WISDM Smart Phone-Based Sensor Mining Architecture," *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data (at KDD-11)*, San Diego, CA
3. GitHub Link To Project Repository

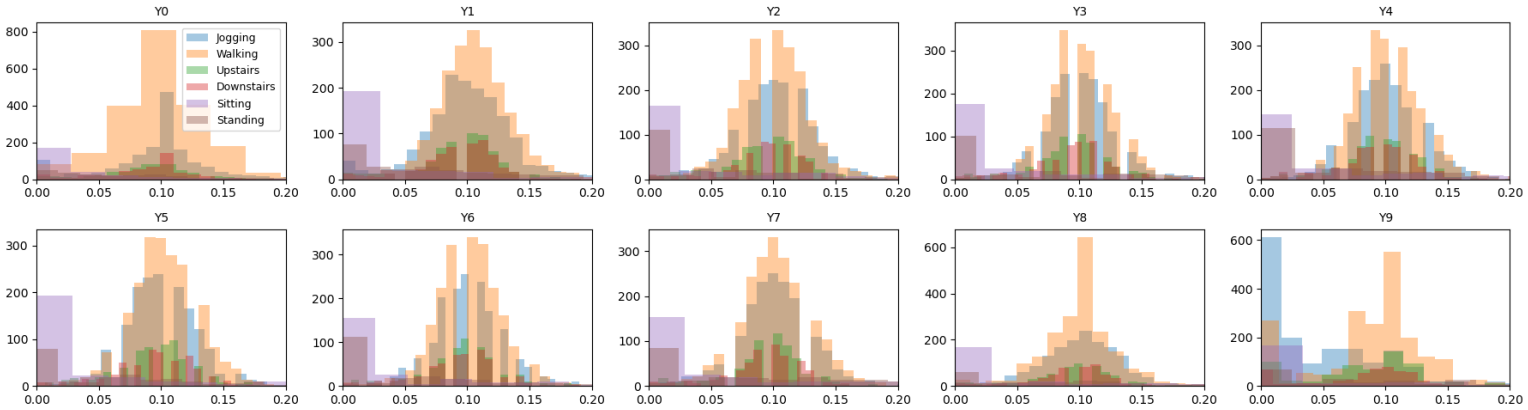
Appendix

(A.)

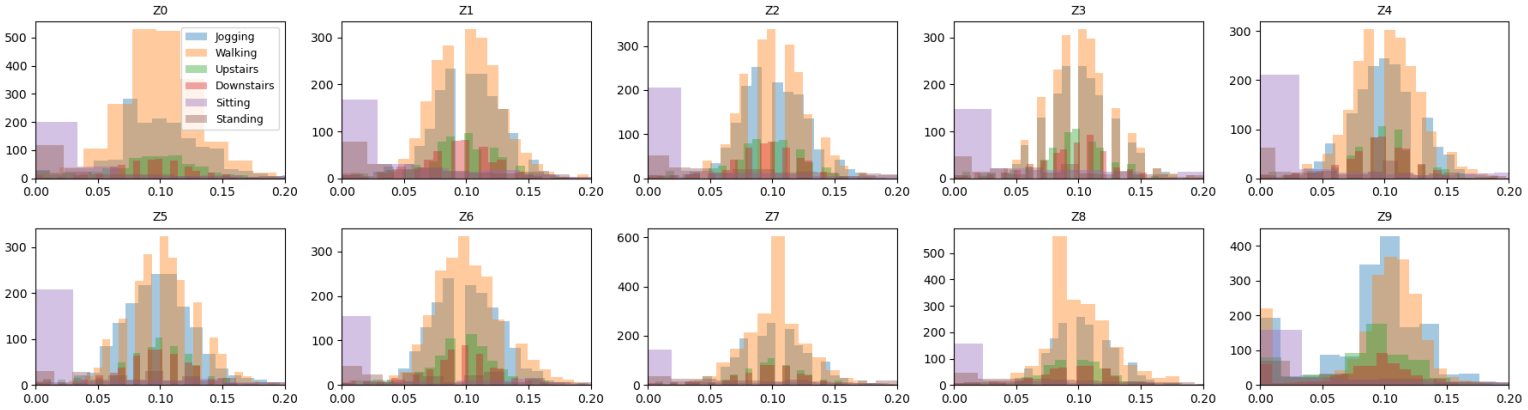
X-Axis Histogram Bins by Activity (X0–X9)



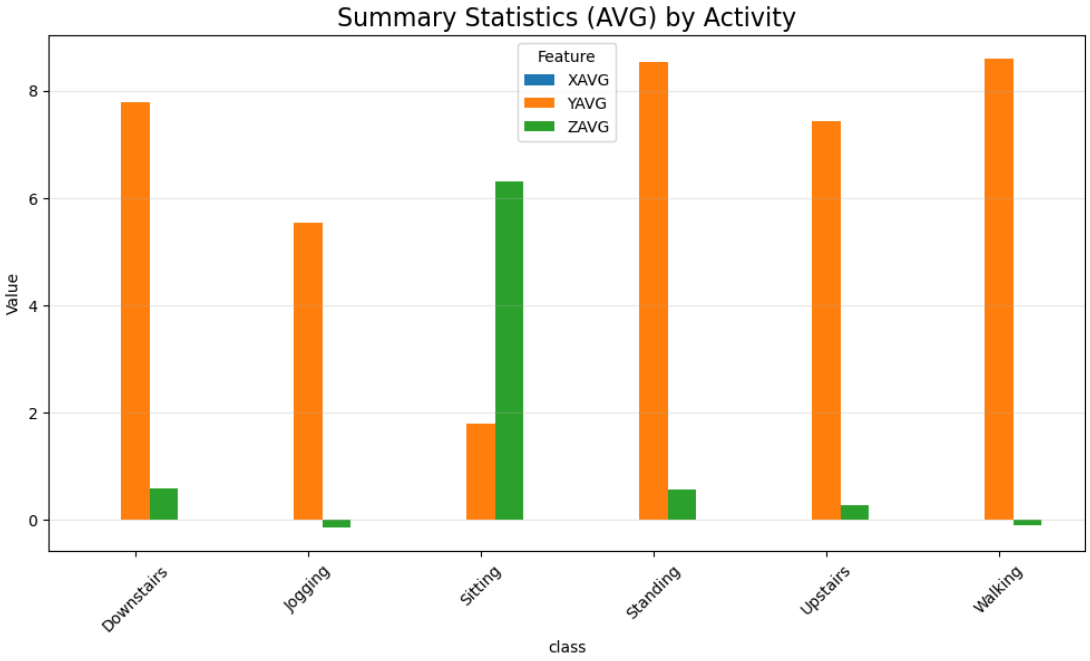
Y-Axis Histogram Bins by Activity (Y0–Y9)

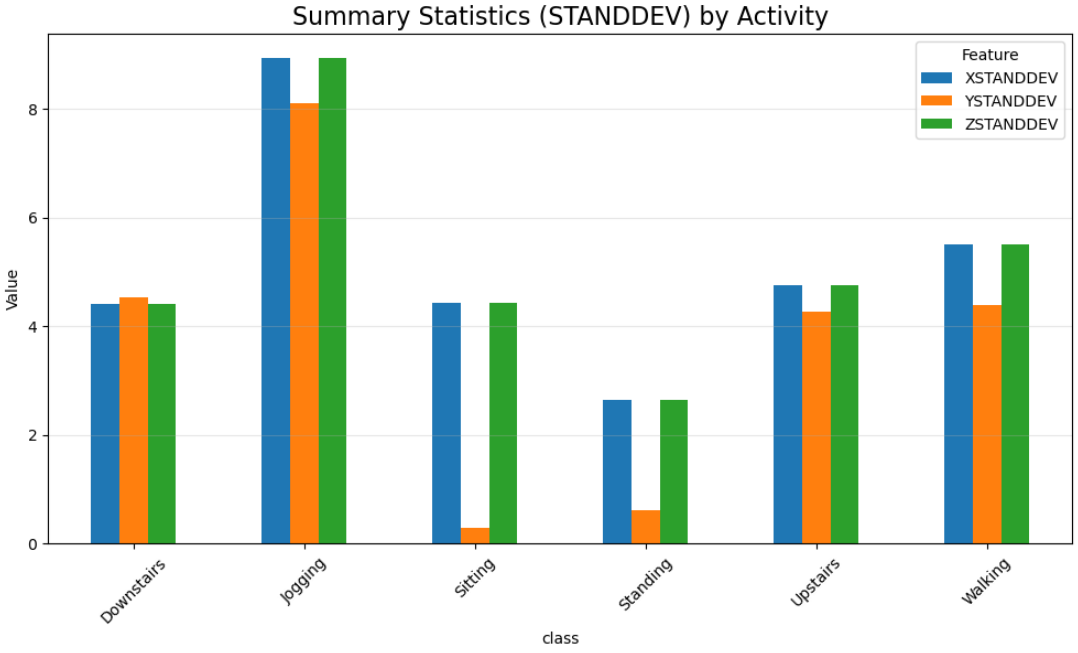


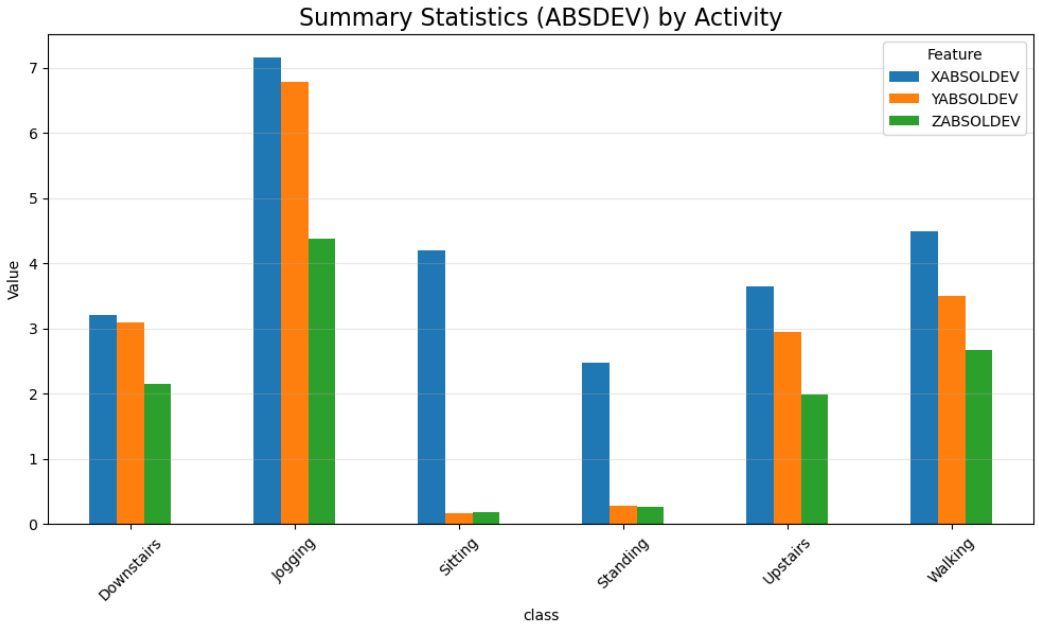
Z-Axis Histogram Bins by Activity (Z0–Z9)



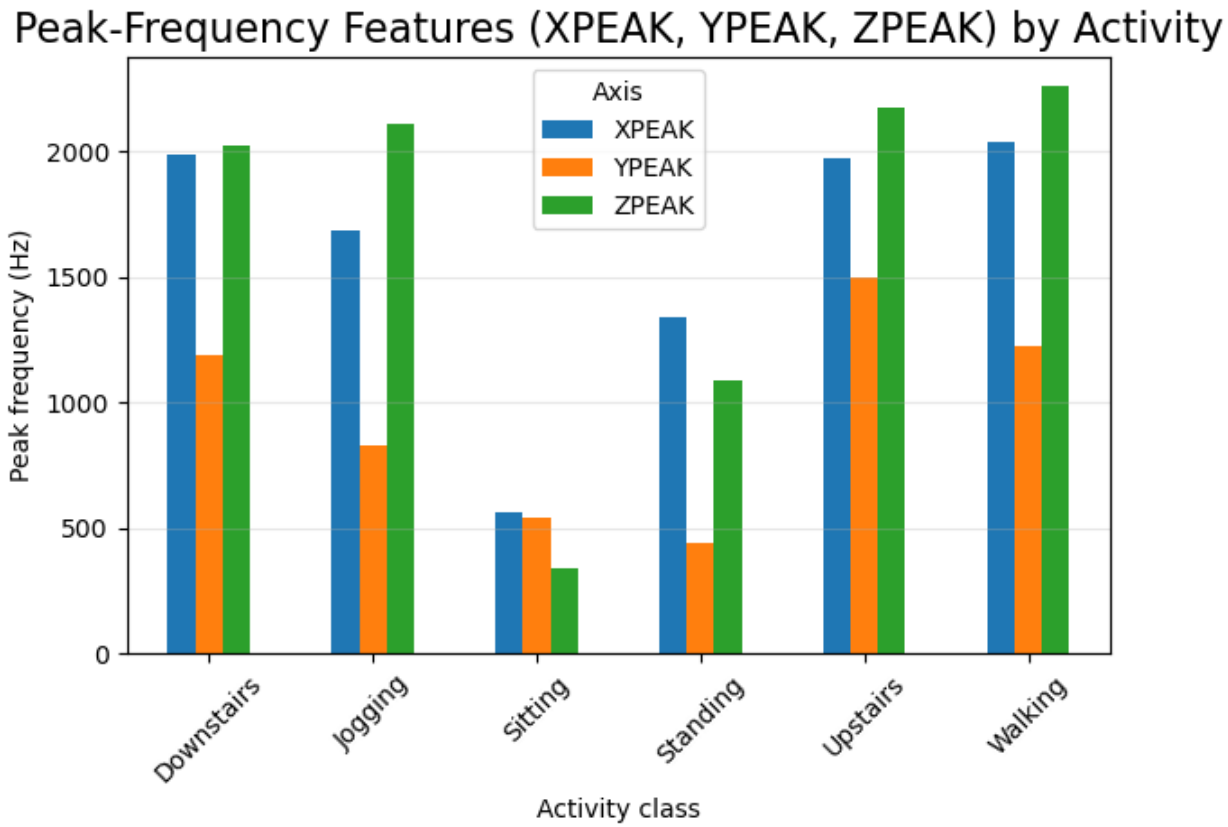
(B.)


Summary Statistics (AVG) by Activity
Feature
XAVG
YAVG
ZAVG
Value
0
2
4
6
8
Downstairs
Jogging
Sitting
Standing
Upstairs
Walking
class


Summary Statistics (STANDDEV) by Activity
Feature
XSTANDDEV
YSTANDDEV
ZSTANDDEV
Value
0
2
4
6
8
Downstairs
Jogging
Sitting
Standing
Upstairs
Walking
class


Summary Statistics (ABSDEV) by Activity
Feature
XABSOLDEV
YABSOLDEV
ZABSOLDEV
Value
0
1
2
3
4
5
6
7
Downstairs
Jogging
Sitting
Standing
Upstairs
Walking
class

(C.)


Peak-Frequency Features (XPEAK, YPEAK, ZPEAK) by Activity
Axis
XPEAK
YPEAK
ZPEAK
Peak frequency (Hz)
0
500
1000
1500
2000
Downstairs
Jogging
Sitting
Standing
Upstairs
Walking
Activity class