Group 15: Classification of Cyclic vs. Non-Cyclic Lower Limb Motion

Myra Forester
College of Engineering
Northeastern University
Boston, Massachusetts
forester.m@northeastern.edu

Himaja Reddy Ginkala
Khoury College of Computer Sciences
Northeastern University
Boston, Massachusetts
ginkala.h@northeastern.edu

Fatima Mumtaza Tourk
College of Engineering
Northeastern University
Boston, Massachusetts
tourk.f@northeastern.edu

Abstract*—Assistive exoskeletons benefit myriad users, from the elderly and differently-abled to workers who engage in repetitive motions or heavy-lifting. Control of the exoskeleton, however, is essential for optimizing its benefit to the user. There are an infinite number of possible movements that a user can engage in. The focus of this paper is cyclic lower limb motion. Categorization of cyclic versus non-cyclic movement is a foundational requirement for the work that would lead to adjustments and improvements to assistive exoskeletons. Here, biometric time series data of both cyclic and non-cyclic lower limb motions are used to create a machine learning model designed to make this categorization.

***Keywords*—classification, machine learning, biomechanics, lower limb motion, cyclic motion**

I. Introduction

Assistive exoskeletons have shown promise in the improvement of quality of life for various groups of people. Benefits exist in rehabilitation, walking assistance for the elderly, alleviating the strain that workers who regularly lift heavy loads and objects experience, and providing support for differently-abled individuals. However, exoskeleton control needs to be tuned to the action its wearer is performing in order to provide the optimal torque at the correct time to result in the desired assistance to the motion. While there are numerous tasks that a user could perform, cyclic tasks are the easiest to create control profiles for. This category of tasks classifies those of a repetitive nature and, hence, can be used to describe motions such as walking, running, and biking.

Due to the immeasurable number of motions that humans perform, one control strategy would be to categorize actions as cyclic versus non-cyclic. The ability to recognize a motion as cyclic would aid in creating predetermined assistive profiles for such motions, enabling non-assistive modes to be used for non-cyclic motions. The task, then, is to effectively classify a given motion as cyclic or non-cyclic.

Because there are infinite actions that a user could perform, it is impossible to generate a vast enough dataset that includes every motion, cyclic and non-cyclic. However, since there are a limited number of cyclic motions, training on a cyclic dataset and classifying non-cyclic motion as outliers could be a potential method to work with limited data to correctly classify motion as cyclic or non-cyclic. This study uses a biomechanics dataset that includes the position, velocity, and acceleration of different body parts over time during various motions in an attempt to implement this strategy of classifying cyclic versus non-cyclic movement from a training database of cyclic motion.

II. Related Work

Research in the study of assistive exoskeletons has shown the potential for a machine learning integrated approach to enhance performance for the user.

A. Real-Time Neural Network-Based Gait Phase Estimation Using a Robotic Hip Exoskeleton

A particular state variable that is crucial for the exoskeleton controller to provide the user with accurate assistance is known as gait phase. Conventional methods estimate gait phase by calculating average stride time, but this approach falls short when adapting to dynamic speeds. Kang et. al developed a sensor fusion-based neural network model to estimate the gait phase in real-time, adaptable to dynamic speeds ranging from 0.6 to 1.1 m/s. Analysis of this methodology was done by studying a group of ten able-bodied subjects walking with an exoskeleton using the estimator, along with provision of the corresponding torque assistance. The model resulted in 36.0% reduction in estimation error and 40.9% reduction in torque error compared to conventional methods. The researchers discovered that a general user-independent model trained on user-specific data outperforms the user-specific model and user-independent model. It was concluded that a sensor fusion-based machine learning model can accurately estimate a user's gait phase and, therefore, improve the controllability of a lower limb exoskeleton. [1] While this study focused on the estimation of gait phase, which is only applicable in cyclic motions, it highlights the importance of an overall controller to indicate to the exoskeleton when it is necessary to use these cyclic control modes as opposed to non-cyclic control modes which are not based off of gait phase.

B. Human Motion Classification Based on Multi-Modal Sensor Data for Lower Limb Exoskeletons

Based on previously-defined motion patterns, researchers utilized Hidden Markov Models to create a classification system to allow for the categorization of multi-modal sensor data acquired from a lower limb exoskeleton. The training set was collected from a total of 10 subjects performing 13 different motions with a passive exoskeleton equipped with seven 3D-force sensors and three inertial measurement units (IMUs). The evaluation allowed for some corrective latency, a validation process for a training set containing all 10 subjects, and a leave-one-out validation process to assess the model's performance. The research group saw an accuracy of 92.80% pertaining to the model's classification ability for the subjects' motions that were included in the training set. [2] However, this study did not generalize to motions that were not included in the dataset, limiting its use to actions it has been trained on. The aim of this study is to create a model that can classify motions it has not been trained on.

III. Data

Classification of whether an action is cyclic (class 1) or not (class 0) will be made using biometric time series data. The biometric data used to build the model is similar to that which an exoskeleton would have access to from its sensors.

A. Repository

A repository of data from Northeastern University's Shepherd Lab was used. It is comprised of data for a combination of cyclic and non-cyclic activities: the 33 cyclic activities consist of variations of walking, shuffling, skipping, squatting, and climbing stairs; the 40 non-cyclic activities include ball tosses, jumps, weight lifts, lunges, tug of war, and numerous other singular or unpredictable motions.

B. Pre-processing

Some of the data was sparse and had limited or missing values for certain activities, so only measurements that were more completely recorded were used. These measurements included the following categories: limb position quantification measurements, including hip flexion, hip adduction, hip rotation, knee angle, ankle angle, and subtalar angle; inertial measurement unit values, including shank, thigh, and pelvis accelerometer and gyroscope readings; and velocity measurements, including hip flexion velocity, hip adduction velocity, hip rotation velocity, knee velocity, ankle velocity, subtalar velocity.

Some of each activity's biometric data was collected every 0.0005 seconds, while other data was collected every 0.005 seconds. In order to maintain a dense dataset while being able to analyze each feature simultaneously, only time intervals where all data was collected were used. This reduced the amount of data by a factor of 10, of which 9/10ths would have been incomplete.

Since human movement does not typically happen at a pace above 10 Hz, the models were trained using a downsampled dataset. The downsampled datasets reduced the number of datapoints used – down to one every 0.1 seconds (as opposed to 0.005 seconds, or 200 Hz). The accuracy of the models improved after this change (see Results section).

Additionally, a multivariate time series dataset was created by combining the information stored in multiple data points. This was accomplished by creating new data points that combined the previous three data points by merging them into one vector (with three times the number of original features). The number of datapoints, then, was reduced by a factor of three, and each represented the passage of 0.3 seconds. This also improved the overall accuracy of the models (see Results section).

IV. Methodology

A. Support Vector Machines

In the training phase, the one-class SVM is provided with a dataset that contains only instances from the normal class, which for this project was implemented using the scikit-learn OneClassSVM module. It learns to find a hyperplane that best separates the normal data points from the origin (center) of the feature space. The origin represents the region where outliers would likely be located. The one-class SVM aims to find the optimal separating hyperplane that maximizes the margin (distance) between the hyperplane and the closest normal data points. In our implementation, scikit-learn DBSCAN clustering algorithm was used to cluster the data, with outliers being points that are not included in any of the clusters. It ensures that all the normal data points lie on one side of the hyperplane, while the outliers lie on the other side. To handle non-linearly separable data, the one-class SVM employs the "kernel trick." This technique allows the algorithm to implicitly map the input data into a higher-dimensional feature space, where the data points might become linearly separable. In this study, a linear kernel, a Radial Basis Function (RBF) kernel, and a Polynomial kernel were each used. During the testing or prediction phase, new data points are evaluated based on their position relative to the separating hyperplane. If a new data point lies on the same side as the majority of the training data (normal class side), it is considered an inlier. Otherwise, if it falls on the opposite side (away from the majority of the training data), it is classified as an outlier or anomaly.

B. Local Outlier Factor

For each data point in the dataset, Local Outlier Factor (LOF) determines its neighborhood by identifying the k-nearest neighbors, implemented from the sklearn-neighbors module LocalOutlierFactor. The value of 'k' is a hyperparameter chosen by the user (our team used 5 after some experimentation) and represents the number of neighbors to consider when calculating the local density of a point. LOF calculates the local density of each data point based on the distance to its k-nearest neighbors. The local density of a point is inversely proportional to the average distance between the point and its neighbors. A high local density means the data point has many neighbors nearby, indicating that it lies within a densely populated region.

For each data point, LOF compares its local density to the local densities of its neighbors. If a point has significantly lower local density than its neighbors, it is considered to be in a sparser region, and its LOF score will be high. The LOF score for each data point is computed by comparing its local density with the local densities of its neighbors. Specifically, the LOF score of a data point is the ratio of the average local density of its neighbors to its own local density. A high LOF score implies that the point is an outlier, as it is less dense compared to its neighbors. To identify outliers, a threshold value (0 for this team's code) for the LOF score is chosen. Data points with LOF scores greater than the threshold are considered outliers, while those with scores below the threshold are classified as inliers. Another important parameter is the contamination factor, where the expected percentage of the test dataset that is in the outlier class is specified.

C. Convolutional Neural Network

Time series data is represented as a sequence of data points ordered by time. Each data point represents multiple values (multivariate time series) at a specific time step. A 1D convolutional layer consists of a set of learnable filters that slide along the time axis of the input sequence, capturing local patterns and features. Each filter's weights are learned during the training process to detect relevant temporal patterns. After the 1D convolution, an activation function (in our example, we used RELU) is applied element-wise to introduce non-linearity and enhance the model's ability to learn complex temporal dependencies.

Pooling layers are also used in this implementation of the 1D-CNN. Max pooling is applied to reduce the length of the sequence by selecting the maximum value in a local region. Pooling helps in reducing the computational complexity and controlling overfitting. After several 1D convolutional and pooling layers, the resulting feature maps are flattened and passed through fully connected layers. These layers process the extracted features and learn higher-level representations of the time series. The final layer of the 1D-CNN produces the network's predictions based on the specific task. For time series forecasting, the output layer can be a fully connected layer with a single output node representing the predicted value at the next time step. For our binary classification task, the output layer uses sigmoid activation. The 1D-CNN is trained using labeled time series data, and the Adaptive Moment Estimation optimization technique is used to minimize the defined loss function. Backpropagation is employed to update the model's parameters during training.

D. Cross-validation

One commonly used statistical method used to estimate the skill of machine learning models is k-fold cross validation. This procedure is used to estimate the skill of a model on new data. For example, a 10-fold cross validation divides a dataset into 10 randomly-selected folds. During a run, each fold has a turn as the test set, while the other folds together become the training or validation sets. This process is repeated 10 times, such that each fold has a turn as the test set. The average error of all the runs is the total error. This technique prevents overfitting and additionally prevents any variance in the sampling from adversely affecting the model.

E. Ablation

The removal of a component from a learning system is known as ablation. Hence an ablation study investigates the performance of a system by removing certain components in order to understand the contribution of those particular components to the overall system. In the context of a machine learning model, methods of exploration include removal of particular variables, removal of parts of the original model, downsampling the data set, etc.

V. Experiments and Results

A. Support Vector Machines

In all instances, the model was trained only on cyclic data with class label 1. During testing, it was evaluated on both cyclic data (positive class, label 1) and non-cyclic data (majority class, label 0). The positive class (cyclic data with label 1) is the class for which the model was trained.

The first method our group tried was the One-Class support vector machine method. Our first instance involved trying the One-Class SVM on the original dataset (Table 1). The model performs reasonably well on the majority class (label 0.0, or non-cyclic data), with a high precision (0.93) and recall (0.65), indicating that it correctly identifies a good proportion of true negatives. However, the model struggles to identify the positive class (label 1.0, or cyclic data), as indicated by low precision (0.03) and recall (0.21), and the high amount of false positives, as shown in the confusion matrix. The F1-score is a harmonic mean of precision and recall and is relatively low at 0.06 for the positive class.

```
Accuracy: 0.63
              precision    recall  f1-score   support

         0.0       0.93      0.65      0.77    303394
         1.0       0.03      0.21      0.06     17608

    accuracy                           0.63    321002
   macro avg       0.48      0.43      0.41    321002
weighted avg       0.88      0.63      0.73    321002
```

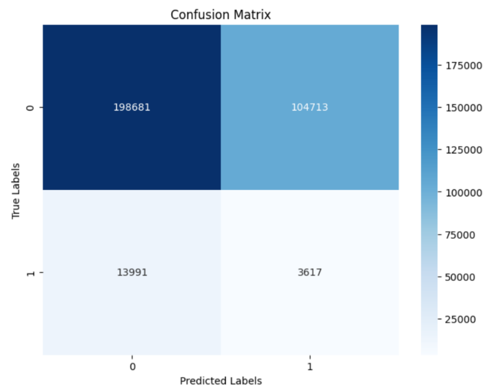


Table 1. One-Class SVM run on original dataset, accuracy table and confusion matrix

Our second instance involved running the One-Class SVM on the downsampled dataset, as mentioned in the preprocessing section. The model's accuracy has improved on the downsampled dataset compared to the original one (Table 2). The precision (0.94) and recall (0.90) for the majority class (non-cyclic data) are high, indicating good performance, and the amount of true negatives were high. However, the model still struggles to identify the positive class (label 1.0, or cyclic data) with low precision (0.02) and recall (0.03), resulting in a very low F1-score (0.02) for the positive class.

```
Accuracy: 0.85
              precision    recall  f1-score   support

         0.0       0.94      0.90      0.92     15204
         1.0       0.02      0.03      0.02       885

    accuracy                           0.85     16089
   macro avg       0.48      0.47      0.47     16089
weighted avg       0.89      0.85      0.87     16089
```

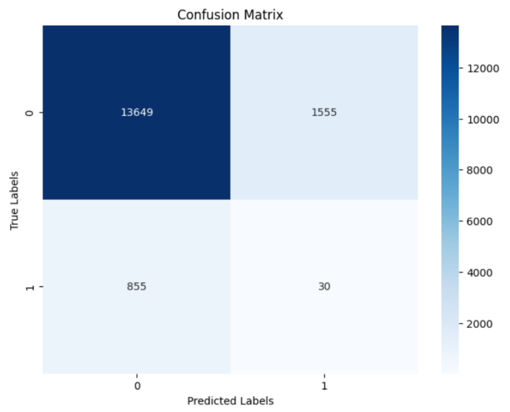


Table 2. One-Class SVM run on downsampled dataset

The third instance involved adding in time information to the model by making each point of the dataset a vector of 0.3 seconds of the original points of the dataset. The model's accuracy has improved even further for the negative class when time information is added to each data point (Table 3). The precision (0.94) and recall (0.97) for the majority class are high, indicating excellent performance. However, the model still struggles to identify the positive class (label 1.0, or cyclic data) with low precision (0.01) and recall (0.00), resulting in a very low F1-score (0.01) for the positive class. While the false positives were improved, the true positives decreased and the false negatives stayed roughly the same.

```
Accuracy: 0.92
              precision    recall  f1-score   support

         0.0       0.94      0.97      0.96     15124
         1.0       0.01      0.00      0.01       871

    accuracy                           0.92     15995
   macro avg       0.48      0.49      0.48     15995
weighted avg       0.89      0.92      0.91     15995
```

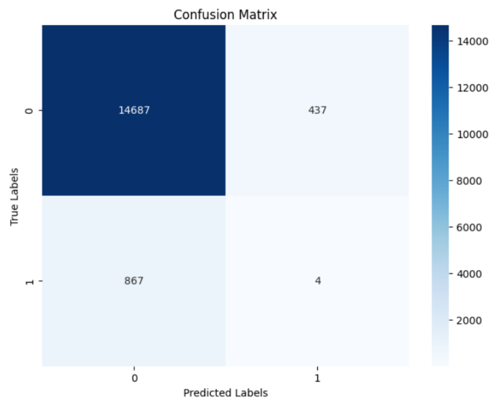


Table 3. One-Class SVM run on downsampled and time incorporated dataset, linear kernel

The fourth instance involved switching the kernel to an RBF kernel instead of the linear kernel. The model's accuracy has significantly improved with the RBF kernel on the time dataset (Table 4). It performs well in identifying the majority class (label 0.0) with high precision (0.95) and recall (1.00). This high accuracy is a bit of a false lead however, as although the false positives were reduced to 0, the model did not predict any positive class instances.

```
Accuracy: 0.95
              precision    recall  f1-score   support

         0.0       0.95      1.00      0.97     15124
         1.0       0.00      0.00      0.00       871

    accuracy                           0.95     15995
   macro avg       0.47      0.50      0.49     15995
weighted avg       0.89      0.95      0.92     15995
```

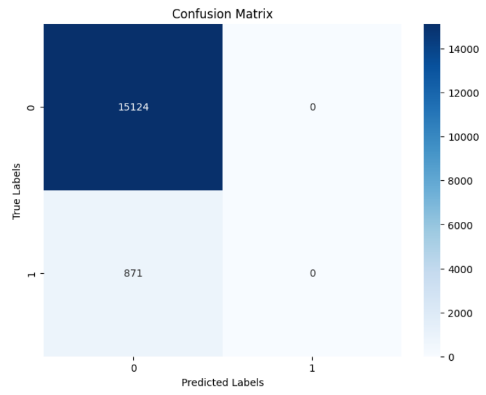


Table 4. One-Class SVM run on downsampled and time incorporated dataset, RBF kernel

The next instance involved switching to a polynomial kernel. The model's accuracy has decreased when using the polynomial kernel on the vectorized dataset (Table 5). While it still performs well in identifying the majority class (label 0.0) with high precision (0.94) and recall (0.95), it did not correctly identify any of the positive class.

```
Accuracy: 0.89
              precision    recall  f1-score   support

         0.0       0.94      0.95      0.94     15124
         1.0       0.00      0.00      0.00       871

    accuracy                           0.89     15995
   macro avg       0.47      0.47      0.47     15995
weighted avg       0.89      0.89      0.89     15995
```

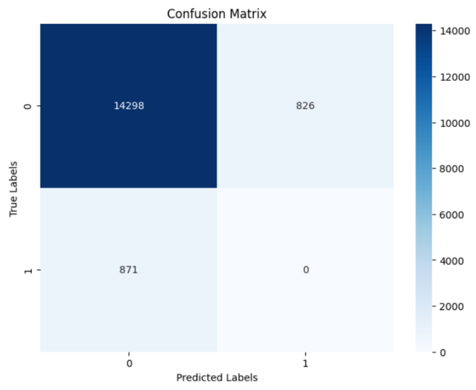


Table 5. One-Class SVM run on downsampled and time incorporated dataset, polynomial kernel

B. Local Outlier Factor

Since the One-Class SVM was not successful at predicting the positive class (cyclic actions) despite having high overall accuracy, the team switched methods to the Local Outlier Factor (LOF) method. Our team first tried the LOF method with a contamination factor of 0.1 The model performed perfectly on the training set when only the cyclic data is present (Table 6). The model achieves a very low accuracy of 0.15 on the test set, indicating that it struggles to generalize to the test data.
For the majority class (label 0.0), the precision is high (1.00), but the recall is very low (0.11), indicating that the model identifies only a small portion of true negatives (correctly identifying non-cyclic actions). For the positive class (label 1.0), the precision is very low (0.06), and the recall is high (1.00), indicating that the model identifies almost all true positives (correctly labeling cyclic actions) but also many false positives (labeling non-cyclic actions as cyclic actions).

```
Train set (cyclic data only):
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         0
         1.0       1.00      0.90      0.95      4167

    accuracy                           0.90      4167
   macro avg       0.50      0.45      0.47      4167
weighted avg       1.00      0.90      0.95      4167

Test set:
              precision    recall  f1-score   support

         0.0       1.00      0.11      0.19     15124
         1.0       0.06      1.00      0.11       871

    accuracy                           0.15     15995
   macro avg       0.53      0.55      0.15     15995
weighted avg       0.95      0.15      0.19     15995
```

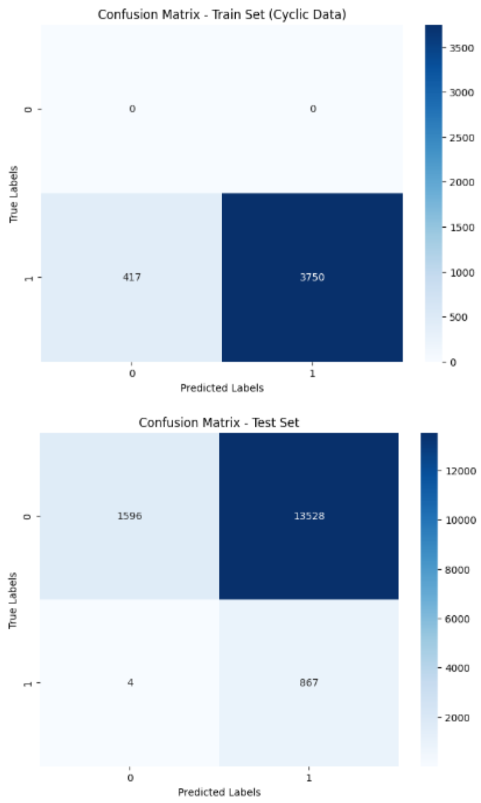
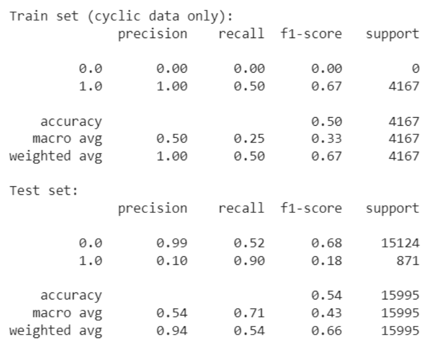
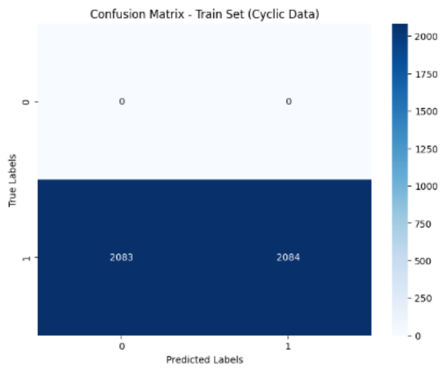


Table 6. LOF, contamination factor of 0.1

When the team increased the contamination factor to 0.5, the highest the model allows, the model also performed perfectly on the training set when only the cyclic data is present, and the model achieved a very low accuracy of 0.54 on the test set (Table 7). While this was one of the highest number of true negatives identified, the model only had approximately 50% accuracy in the negative class. Changing the number of n-neighbors also did not help the accuracy or f1 score.



```
Train set (cyclic data only):
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         0
         1.0       1.00      0.50      0.67      4167

    accuracy                           0.50      4167
   macro avg       0.50      0.25      0.33      4167
weighted avg       1.00      0.50      0.67      4167

Test set:
              precision    recall  f1-score   support

         0.0       0.99      0.52      0.68     15124
         1.0       0.10      0.90      0.18       871

    accuracy                           0.54     15995
   macro avg       0.54      0.71      0.43     15995
weighted avg       0.94      0.54      0.66     15995
```



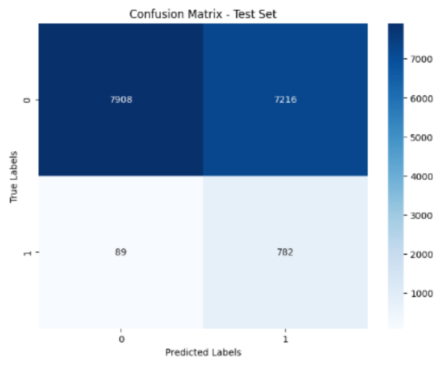
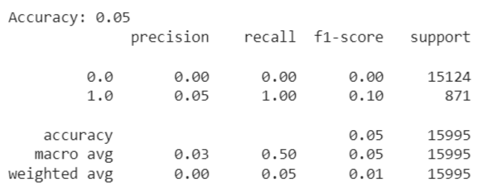


Table 7. LOF, contamination factor of 0.5

C. Convolutional Neural Network

While the LOF performed better than the One-Class SVM at identifying the positive class, there was still a lot of room for improvement in the negative class, which resulted in the team testing CNNs. However, the CNN model achieved a very low accuracy on the time dataset (Table 8). It performs well in identifying the positive class (label 1.0) with high precision (0.05) and recall (1.00). However, it did not identify any of the negative class, indicating that it simply categorized all motions as cyclic. The team tried many different parameters on the CNN including number of epochs, the number of filters used in the convolutional layer, the kernel size, the number of dense units, the batch size, and the number of epochs, but none of these changed the accuracy or f1 score for the CNN.



```
Accuracy: 0.05
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00     15124
         1.0       0.05      1.00      0.10       871

    accuracy                           0.05     15995
   macro avg       0.03      0.50      0.05     15995
weighted avg       0.00      0.05      0.01     15995
```

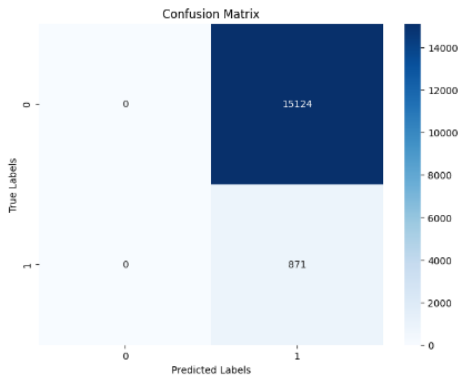


Table 8. Convolutional Neural Network

D. Cross-validation

A 10-fold cross validation was performed for each of the three machine learning models. The evaluation was first done on the original data with the DBSCAN and One-Class SVM model, resulting in an across the board accuracy of 0.39 as can be seen in the figure below. Validation was then done on the downsampled, vectorized data set once again using the DBSCAN and One-Class SVM model, and once again resulting in an accuracy score of 0.39 for each fold.

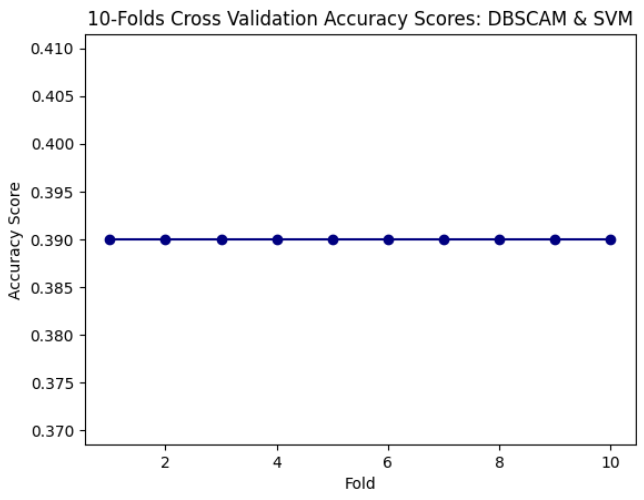


Figure 1. Accuracy Scores for 10-folds, DBSCAN and One-Class SVM

The 10-fold cross validation was then performed using the Local Outlier Factor model on the downsampled, vectorized data. As can be seen below, while there were slight changes in the accuracy scores for the test sets throughout the folds, fluctuating between 0.24 and 0.27, the scores for the train sets remained the same at a score of 0.99.

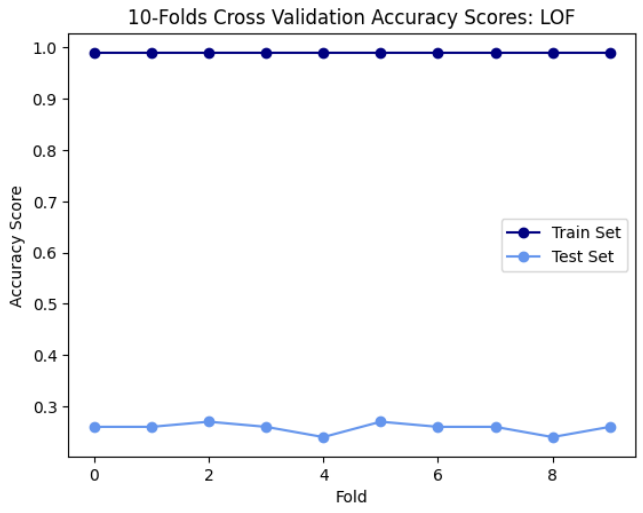


Figure 2. Accuracy Scores for 10-folds, LOF (n = 40, c = 0.1)

When evaluated on the Convolutional Neural Network model with the downsampled, vectorized data, the cross validation resulted in small differences between each fold, but maintained a high level of accuracy scores with the lowest being 0.96. The slight fluctuations of the accuracy can be seen in the figure below.

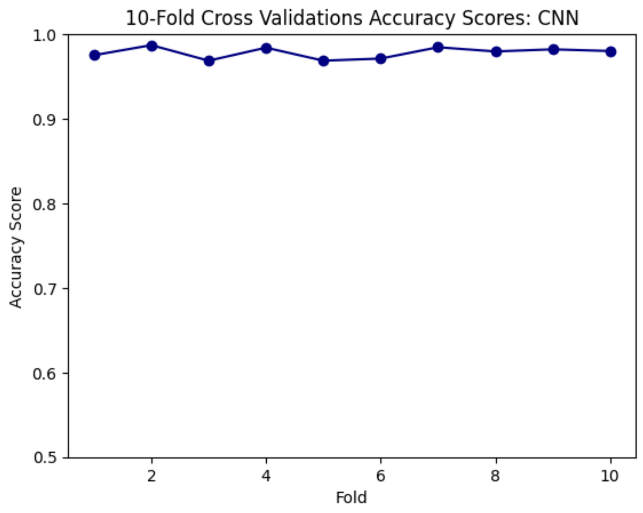


Figure 3. Accuracy Scores for 10-folds, CNN

E. Ablation

Three types of methods were applied for the ablation study for the three machine learning models: removal of particular variables from the original data, downsampling of the original data, and removal of particular variables in the downsampled data.

Firstly, the removal of various variables was experimented with: hip flexion, hip rotation, knee angle, knee velocity and ankle angle. The DBSCAN and One-Class SVM model was run repeatedly by removing one variable at a time while keeping all the others. For example, when removing a variable such as hip flexion, the right and left measurements were removed and all other variables were kept the same. While the removal of most of the variables resulted in either a lower accuracy score or the same as the original, the removal of knee angle resulted in a slightly higher score, recall and f1 score.

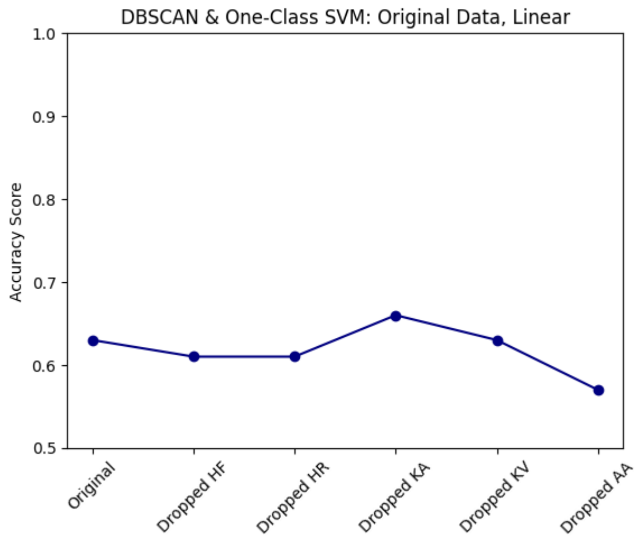


Figure 4. Ablation Accuracy Scores for DBSCAN and One-Class SVM with Linear Kernel

Secondly, downsampling was performed which showed a tremendous improvement in accuracy when using the DBSCAN and One-Class SVM model with a linear kernel, bringing the score from 0.63 to 0.85. The downsampled data was then vectorized with the previous two measurements for the variables. Hence, when a variable such as hip flexion was removed, its corresponding two previous measurements were also removed in order to maintain consistency. Utilizing the downsampled and now vectorized data resulted in another improvement of accuracy. With the DBSCAN and One-Class SVM model, using a RBF kernel, the accuracy score went up to 0.95. Continuing with the experimentation with the SVM model, through ablation, the variables were removed and tested with various SVM kernels, however no significant change in the results was observed with the linear and polynomial kernels. The results of the three various kernels can be seen in the figure below.

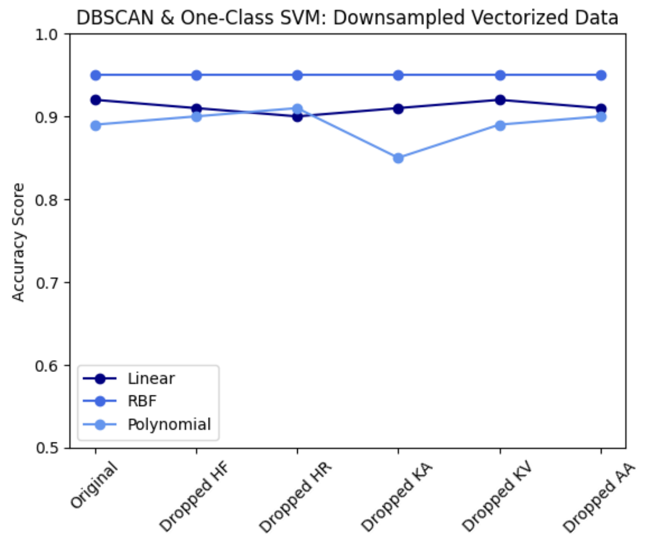


Figure 5. Ablation Accuracy Scores for DBSCAN and One-Class SVM with Linear Kernel, Downsampled and Vectorized Data

The downsampled, vectorized data set was used for the ablation study for the two configurations of the LOF model. As can be seen, for the first configuration, the train set maintained its score of 0.90, and the test set across the board resulted in a score of 0.15.

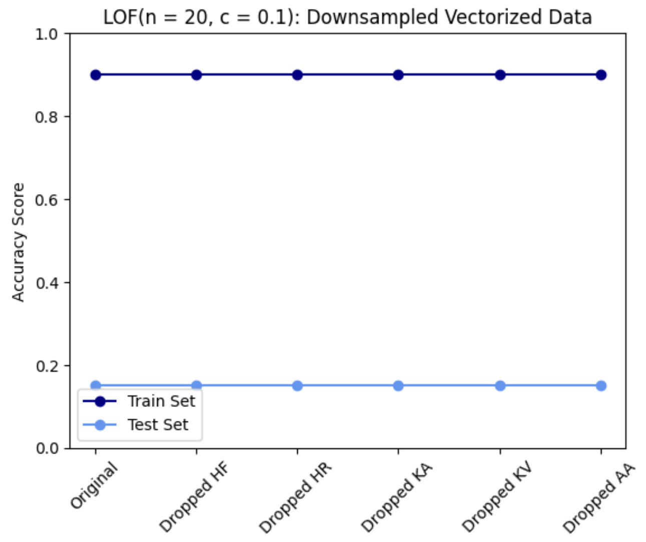


Figure 6. Ablation Accuracy Scores for LOF (n = 20, c = 0.1), Downsampled and Vectorized Data

For the second configuration of the LOF model, the train set showcases a score of 0.50, while the test set score fluctuates between 0.51 and 0.52.

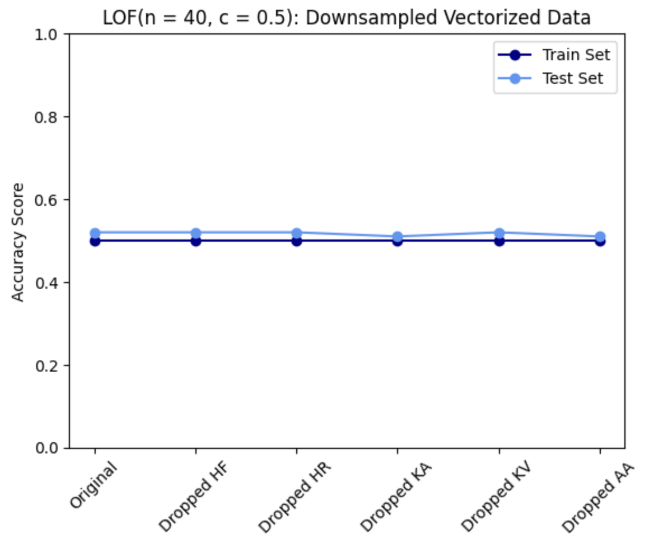


Figure 7. Ablation Accuracy Scores for LOF (n = 40, c = 0.5), Downsampled and Vectorized Data

Four different configurations of the CNN model were tested while also removing variables one at a time. The results across the board provided an accuracy score of 0.05.

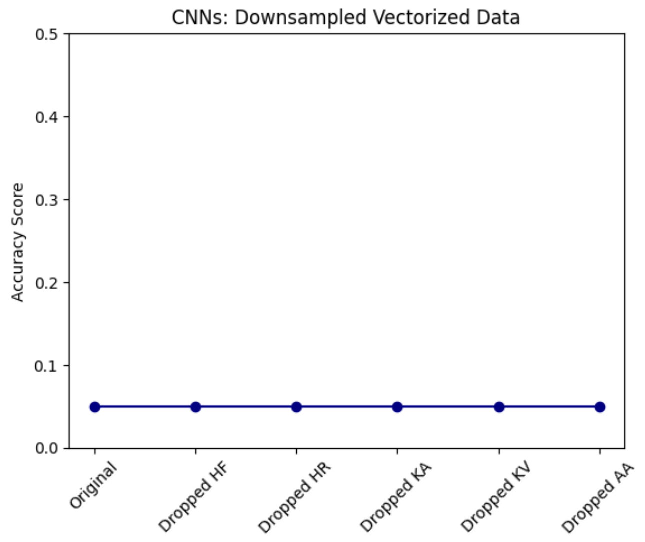


Figure 6. Ablation Accuracy Scores for all CNN configurations

Overall, the ablation study showed that the removal of various variables did not have significant impact, and neither did changes in the configurations of the SVM, LOF and CNN models. It did however showcase that the largest impact on the accuracy rates of the models was caused by the downsampling of the original data, and the an additional improvement with vectorizing the data and using the DBSCAN and One-Class SVM model with a RBF kernel.

VI. Discussion and Conclusion

None of the models had adequate performance for our objective of classifying cyclic and non-cyclic action based on a training dataset of only cyclic data. While this group was able to achieve some improvement to the accuracy by downsampling the data and adding in a time component to each point of the dataset, all of the models failed to accurately classify cyclic vs non-cyclic data.

The SVMs were very successful at classifying non-cyclic data but did not classify cyclic data with high accuracy, with some kernels classifying all motion as non-cyclic. The LOF models, on the other hand, had better accuracy on correctly classifying cyclic motion but terrible accuracy classifying non-cyclic motion. In addition, it is dependent on the parameter of the contamination factor, which is an estimate of the distribution of classes in the test set. This is something that an exoskeleton control system would not have in the real world. The CNN simply classified all motion as cyclic, which was also not suitable for our task.

Our group learned some valuable information about solving this problem including dataset preprocessing and the strengths and weaknesses of each machine learning model for this problem. Some methods had very high accuracy but did not predict the positive class at all, as in the One-Class SVM method with the RBF kernel. Some methods had a higher f1 score for the positive class but lower overall accuracy, as in the LOF model. While the group tried various parameters in each of these methods, the problem remains too complex for the classical machine learning methods. The fact that we could only train on cyclic data as well as the fact that cyclic and non cyclic data have many of the same biomechanical signals, makes this problem very difficult to solve. To accurately solve this problem, the method would need high accuracy in classifying both non cyclic and cyclic actions in the test dataset, as an exoskeleton would have to use different control parameters for each of those scenarios. If this group were to do future work, we would attempt deep learning methods and reinforcement learning in order to address the complexities of this problem.

References

[1] I. Kang, P. Kunapuli And A. J. Young, "Real-Time Neural Network-Based Gait Phase Estimation Using A Robotic Hip Exoskeleton," In Ieee Transactions On Medical Robotics And Bionics, Vol. 2, No. 1, Pp. 28-37, Feb. 2020, Doi: 10.1109/Tmrb.2019.2961749.

[2] J. Beil, I. Ehrenberger, C. Scherer, C. Mandery And T. Asfour, "Human Motion Classification Based On Multi-Modal Sensor Data For Lower Limb Exoskeletons," 2018 Ieee/Rsj International Conference On Intelligent Robots And Systems (Iros), Madrid, Spain, 2018, Pp. 5431-5436, Doi: 10.1109/Iros.2018.8594110.