Classification of Physical Activities Using Sensor Data from Smartwatches

Anuj Kumar
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
anujkumar@cs.umass.edu

Sri Sudhamsu Krishna Manne
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
smanne@cs.umass.edu

Abstract

Wearable consumer electronics with a myriad of sensors are becoming increasingly popular. One major application has been physical activity detection and tracking. We examine existing methods of data pre-processing, feature extraction, and classification, and then propose a new feature space and discuss the results of standard classification techniques based on these new features.

1 Introduction and Problem Statement

In the era of fitness tracking, it has become increasingly popular to have wearable technologies keep trace of an individual's physical activity throughout the day. With some fitness trackers tracking sleep and heart-rate patterns to help guide the individual towards better health, and others tracking walking, running and at most a few gymnasium activities of an individual, a dearth of tracking various sports played by the individual is felt. We hope to address this problem.

In recent years, the consumer electronics market has seen a steady rise in devices that carry several types of sensors. Smartwatches in particular, being worn by an individual almost throughout the day, give a plethora of these sensors to research. These extra sensors, like the gyroscope, are needed in making the jump from classifying simple activities like walking, running and sleeping to classifying more complex activities like multiple sports. Being able to track various sports played will enable a more complete picture of the individual's physical activities and aid the fitness tracker better in gauging health. Furthermore, the individual would also wish to be informed on how much time and resources have been spent towards each sport.

Traditional work in this field has been restricted to using regression-based thresholds and then defining these thresholds for various activities. More recently, Machine Learning has been used for classifying the time series data we get from sensors, by deriving statistical features from them, and then feeding it to a classifier. However, these methods are limited in their accuracy.

In this work, we obtain real-life sensor data from various people involved in six different kinds of activities. After data pre-processing steps, we then convert this time series data into the frequency domain, and use various parameter-optimized classifiers to classify the activity into one of Badminton, Basketball, Foosball, Ice-skating, Running and Walking. The first two sports will not have a lot of exactly repetitive actions, whereas the last two will have lots of those. The middle two will have a moderate number of repetitive actions.

We show that it is relatively easier to classify various sports in the frequency space than using statistical features. Using neural networks and random forest classifiers, we show that for a single individual, the results are surprisingly accurate in estimating an activity into one of the six sports. However, what's more interesting is that a more general model of each sport is harder to learn. Due

Submitted as Project Report for CS589 Machine Learning. University of Massachusetts Amherst. Fall 2016.

to the different playing styles of each individual, the activity data will largely look dissimilar. The classification accuracy drops drastically when trying to classify for multiple individuals.

We explore methods to then generalize the classification of the sport regardless of the individual, while at the same time observing that this is not a limitation towards fitness trackers. Since smartwatches, or other fitness trackers, generally belong to one person, and are worn only by that individual throughout the day, classifying a single person's activities into various sports is sufficiently viable.

2 Related Work

The early attempts at inferring activities from accelerometer data were generally through a custom-made device made to be worn at the hip for the express purpose of research. Ravi, et al. (2005)[1] and Bonomi, et al. (2009)[2] used these custom-made devices to collect the data, extracted statistical features like mean, standard deviation, and cross-correlation, and then employed classification methods like decision trees and SVMs, and reported predictive performance in the ranges of 60% to 99% depending on the activity and setting.

With the advent of modern smartphones with accelerometers, studies made use of them as they were easily available and, more importantly, present in pockets of an ever-increasing number of people around the world. Kwapisz, et al. (2010)[3] conducted studies in line with the works of [1] and [2], but for data collected using smartphones. They too relied on statistical features for classification - which involved multilayer perceptron and logistic regression among other techniques - and reported prediction accuracies ranging from 12% to 90% depending on the activity and classification technique.

The last 3-4 years have seen mass adoption of fitness trackers and smartwatches, and in being able to be worn virtually throughout the day they have a huge advantage over any of the other devices, and predictably they have become the research community's primary source of high-resolution personal activity data. Mannini, et al. (2013)[4] used a customized sensor to be worn on the user's wrist, and transformed the time-domain data to frequency-domain to use as features for classification. They report accuracies ranging from 60% to 99% depending on the activity. We take inspiration for our FFT-based techniques from their study.

Zheng et al [5] used a total of 57 statistical features derived over the time series data collected from smartwatches for various subjects, and used an ensemble of many classifiers using majority vote to help in robustness. They achieved accuracies ranging over 15% to 94% for various physical everyday activities like dancing, lying down, running, sitting, and standing.

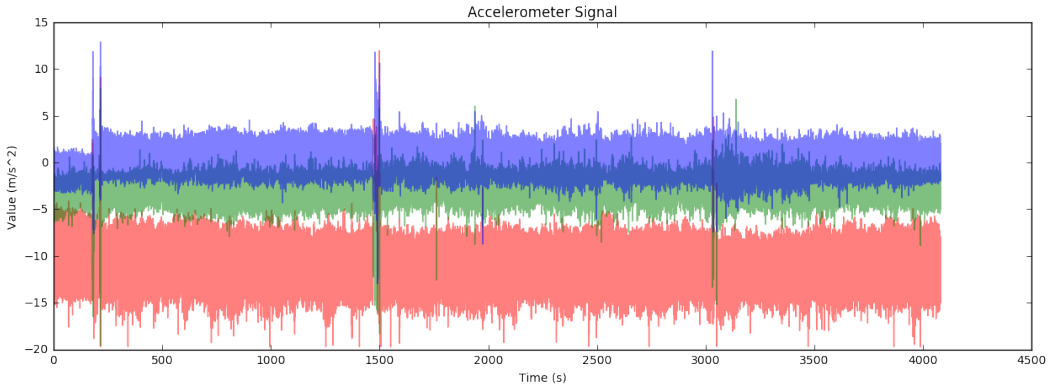
All these studies use only accelerometer data. While accelerations in the three primary axes are a great source of information, we propose that gyroscope data also contains a vast amount of distinguishable information that we can make use of. Another commonality among these studies is that they have concentrated on achieving high classification accuracies among simple repetitive activities such as walking, running, sitting, sleeping, or climbing stairs. We plan to take this further and tackle the problem of complex activity classification like differentiating between sports.

3 Data Set

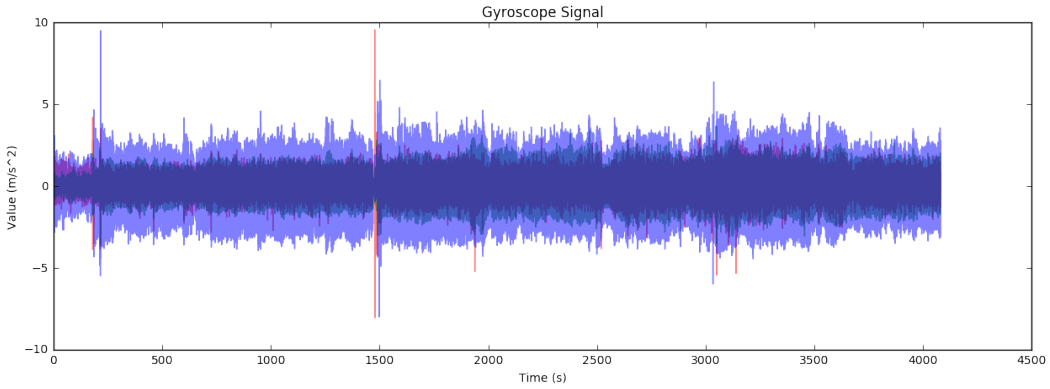
Our data set was manually collected by us since we could not find any relevant data sets online which could be used for the problems we are looking to address. We used the accelerometer and gyroscope sensors from one ASUS ZenWatch 2 WI501Q and one LG G Watch W100 smartwatches through an open source app, Sensor Data Logger [8]. We modified the app to export sensor data of multiple simultaneous sensors - a feature it didn't have before. We also changed the frequency at which the app collects this data. We set the said frequency to 50Hz. Figure 1 shows raw acceleration and gyroscope data for walking over an extended period of time.

The playing data of multiple hours for various people playing each sport was collected through this app. Once we had this raw data, we had to apply various preprocessing and feature extraction steps to be able to feed to classifiers for classification purposes. These are explained below:

The complete data set is divided into three parts. Part A consists of multiple data of a single person (who is not in B and C) in each sport. Part B consists of numerous data of multiple people who are not in data sets A and C. Part C has data of a single person in each sport, this person not contributing



(a) Acceleration signal. Axes colors: X-Red, Y-Green, Z-Blue.



(b) Gyroscope signal. Axes colors: X-Red, Y-Green, Z-Blue.

Figure 1: Raw sensor data for walking over an extended period of time.

towards any of data sets A or B. The use of these three separate cases will become evident when we discuss our experiments. In the pre-processing step, we first trim the data to exclude 15 seconds from either side, thus removing boundary noise which materializes due to various realistic factors as adjusting to the watch after recording is turned on, making another person wear the watch, etc.

We define three parameters: Example Duration (Δ), Filter size (f) and Stride (s). Δ is the length of time which we consider as one example, i.e. one data point for the classification problem. f is the number of seconds before the current time step over which to compute the FFT or statistical features. s is time interval at which we compute the FFT or statistical features for the previous f seconds.

Due to a few inaccuracies in the app, the amount of data for the two sensors may turn out to be slightly unequal for equal durations of time. We make sure to subsample the larger data in such a case to ensure we have the exact same amount of data for both sensors. Once this is done, we derive a complete file encompassing all our examples for a particular data set (A, B or C), while ensuring that the complete file has formatted data in integral multiples of Δ. The classifiers then take these features and labels files as inputs.

3.1 Statistical Features

We build a single training example of our dataset by considering a chunk of size Δ. In this data chunk, we then take smaller blocks of size f separated by stride s. We calculate 16 features over each such block, which corresponds to a column in our feature matrix. We thus create a feature matrix, each entry of which can be thought of as a training image of length 16, width dependent on Δ and s, and height 6 corresponding to each of three accelerometer and three gyroscope channels. Once this feature matrix is completed, we create another matrix, which contains the cross correlation of the six axes (six channels) of each such data block inside a particular chunk. Thus, this particular matrix has entries of dimensions 15 (number of correlations) and width again dependent on Δ and s. In the

Table 1: Statistical Features

1. Mean: $\mu_s = \frac{1}{T}\Sigma_{i=1}^{T} s_i$

2. Standard Deviation: $\sigma_s = \sqrt{\frac{1}{T}\Sigma_{i=1}^{T}(s_i - \mu_s)}$

3. Coefficients of variation: $c_v = \frac{\sigma_s}{\mu_s}$

4. Peak-to-peak amplitude: $max\{s1, ..., sT\}$ - $min\{s1, .., sT\}$.

5-9. Percentiles: 10^{th}, 25^{th}, 50^{th}, 75^{th}, 90^{th}

10. Interquartile range: difference between the 75^{th} and 25^{th} percentiles.

11. Lag-one-autocorrelation: $\frac{\Sigma_{i=1}^{T-1}(s_i-\mu_s)(s_{i+1}-\mu_s)}{\Sigma_{i=1}^{T}(s_i-\mu_s)^2}$

12. Skewness: $\frac{\frac{1}{T}\Sigma_{i=1}^{T}(s_i-\mu_s)^3}{(\frac{1}{T}\Sigma_{i=1}^{T}(s_i-\mu_s)^2)^{3/2}}$, measure of asymmetry of the signal probability distribution.

13. Kurtosis: $\frac{\frac{1}{T}\Sigma_{i=1}^{T}(s_i-\mu_s)^4}{(\frac{1}{T}\Sigma_{i=1}^{T}(s_i-\mu_s)^2)^3} - 3$, degree of the peakedness of the signal probability distribution.

14. Signal power: $\Sigma_{i=1}^{T} s_i^2$

15. Log-energy: $\Sigma_{i=1}^{T} \log s_i^2$

16. Zero crossings: number of times the signal crosses its median.

17. Correlation between each pair of axes: $\frac{\Sigma_{i=1}^{T}(s_i-\mu_s)(v_i-\mu_v)}{\sqrt{\Sigma_{i=1}^{T}(s_i-\mu_s)\Sigma_{j=1}^{T}(v_i-\mu_v)}}$

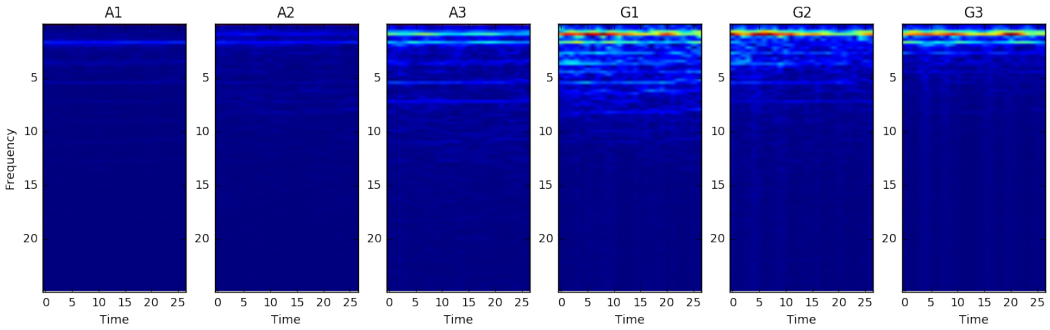


Figure 2: One example from the processed FFT data set for walking. A images are for the three accelerations, G images for the three axes in gyroscope. Colors towards red on the spectrum mean higher power in the FFT. Notice the distinct lines at particular frequencies; those are frequencies at which there is most physical movement during walking.

end, we concatenate the two matrices in 2D to give us a final array which can be used directly for the classifier. Such an array has each row corresponding to one entry of either feature matrix.

The 16 statistical features used to create the feature matrix depicted above are based on Zheng et al. [5]. These features are detailed in Table 1.

3.2 FFT Features

We then wish to convert the data into the Fourier space intelligently. We define one training example on our data set to be over the example duration Δ. In this duration, we take Fast Fourier Transforms of data of size f separated by stride s. We use this Fourier transform as a column vector in our example. Thus, each entry in our feature matrix can be thought of as an image with height equal to size of the fast Fourier transform, width dependent on Δ and s, and with 6 channels corresponding to each of three accelerometer and three gyroscope channels. We then reshape this to two dimensions for our classifiers. Figure 2 shows a sample from the processed FFT data set for walking. Figure 3 shows a similar sample for Badminton; notice the distinct vertical bands of action.

4 Proposed Solution

4.1 Dimensionality Reduction of FFT Features Using PCA

In case of FFT features, we start with Principal Component Analysis (PCA), separately for each sensor channel with whitening and pass these features onto the classifiers. We feel there is a need for PCA because the distinguishable dimensions in the FFT features are sparse, as can be seen in the one

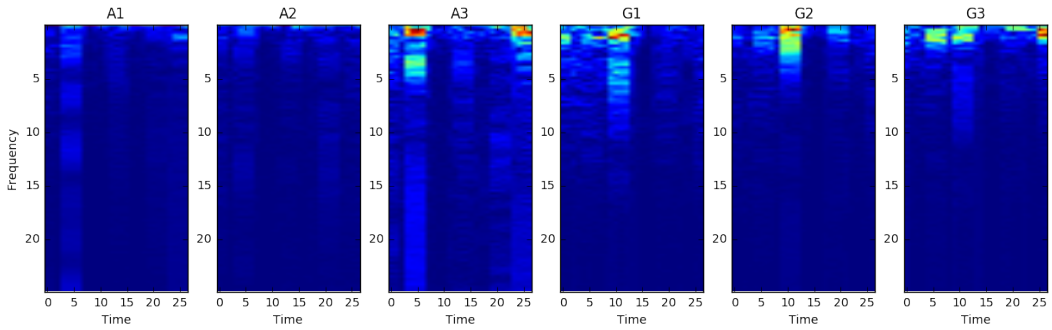


Figure 3: One example from the processed FFT data set for badminton. A images are for the three accelerations, G images for the three axes in gyroscope. Colors towards red on the spectrum mean higher power in the FFT. Notice the distinct vertical lines. These are times at which there was a sudden strong movement like swinging the racket. Also notice the differences when compared to the walking data.

sample of the walking data in Figure 2. After a series of test runs, we chose to keep 100 dimensions per sensor channel in the transformed features. Depending on the example duration, filter size and stride in the example, the original dimensionality can be anywhere from 200 to 3000 per channel.

We do not use PCA for the statistical features because they are already dense and substantially fewer in number. They will be passed on as is to the classifiers.

4.2 Random Forest Classifier

Using scikit-learn libraries [6], we set up a pipeline for a random forest classifier. It involves PCA when FFT features are provided and the classifier itself. Then we perform a grid search employing cross-validation over 5 folds to find the optimal hyperparameters for the classifier which include minimum sample splits, split criterion, and the number of estimators for the random forest classifier and the number of components to keep in case of PCA.

We chose random forest classifier because of the impressive performance shown by them as empirically seen in previous studies.

4.3 Neural Network Classifier

With TensorFlow [7] as the neural network framework, we use a three-layer fully connected neural network with 1024 units in each of the two hidden layers and 6 units in the output layer for the 6 activity classes. Each hidden layer is followed by dropout. The learning rate of the network and the keep probability of the dropout are decided by 4-fold cross-validation. The training is done for 20 epochs during cross-validation, and for 40 epochs after that using the optimal hyperparameters. The sample batch size for the training is kept at 25 in all cases. A softmaxed cross-entropy is the objective to minimize. We wanted our model to have high capacity hence we chose a large layer size of 1024 for the hidden layers. We overcome overfitting using dropout, which can act as a regularizer.

Mathematically, our network can be represented by three weight matrices ($\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{W_3}$) and three bias vectors ($\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$) for the three layers of the network.

$$
\begin{aligned}
\mathbf{h_1} &= \mathbf{X}\mathbf{W_1} + \mathbf{b_1} \\
\mathbf{h_{1,drop}} &= dropout(\mathbf{h_1}, p) \\
\mathbf{h_2} &= \mathbf{h_{1,drop}}\mathbf{W_2} + \mathbf{b_2} \\
\mathbf{h_{2,drop}} &= dropout(\mathbf{h_2}, p) \\
\mathbf{y_{out}} &= \mathbf{h_{2,drop}}\mathbf{W_3} + \mathbf{b_3}
\end{aligned}
$$

The model is then made to minimize the softmaxed cross-entropy loss between $\mathbf{y_{out}}$ and the ground truth labels. p is the hyperparameter for keep probability of the dropout layer. $\mathbf{X}$ is the input data. $\mathbf{h_1}$, $\mathbf{h_2}$, $\mathbf{y_{out}}$ are the activations at each of the affine layer. $\mathbf{h_{1,drop}}$ and $\mathbf{h_{2,drop}}$ are the dropped-out activations from the dropout layers.

Table 2: Performance of Random Forest Classifier on Statistical Features

Data Sets		Meta-hyperparams (sec)			Optimized hyperparams		Accuracies (%)	
Input	2^{nd} Test	Δ	f	s	minSamp*	splitCrit**	Test	2^{nd} Test
A	-	15	4	1	10	gini	58.37	-
A	-	30	2	1	10	gini	50.25	-
A	-	30	4	1	5	entropy	50.25	-
A+B+C	-	15	4	1	5	entropy	56.41	-
A+B+C	-	30	2	1	5	gini	48.58	-
A+B+C	-	30	4	1	10	entropy	58.87	-
A	B+C	30	4	1	5	gini	44.28	**35.48**
A+B	C	30	4	1	5	gini	**62.69**	23.85

*minSamp is the minimum number of samples required in node for tree to split on
**splitCrit is the criterion to split the node on: gini or entropy

5 Experiments and Results

We perform various experiments on our data sets A, B and C taken in various groupings, and report our results below. The motivation for the groupings is to learn a generalized version of each sport instead of a single person specific model. For each of these experiments we split the input data into a training set and a test set in 80/20 ratio. In some cases we also have another exclusive test set with data from people not present in the input training data.

From our experiments, we also noticed that the meta-hyperparameters that define how coarse or fine our FFT-based features are are not dependent on the kind of classifier we are using. It seems that there exists a combination of meta-hyperparameters - Δ, s and f - which perform best and improve accuracy for any classifier. We infer that when we use these meta-hyperparameters, we are essentially imparting the largest amount of distinguishable information about the concerned activities.

5.1 Statistical Features with Random Forest Classifier

The statistical features generated are passed on to the Random Forest Classifier. Unlike FFT features, no PCA is required for in this case, and the hyper-parameter search space has been kept lower in the Table 2 observing the trend in various runs of the algorithm, and fixing optimized parameters. These seem to perform poorly when compared to results from use of FFT features.

5.2 FFT Features with Random Forest Classifier

We use Random Forest Classifier from the scikit-learn library to classify our data with FFT-based features. The results are shown in Table 3. For each combination of Δ, s and f, we searched over a widely varying hyper-parameter space and employed 5-fold crossvalidation using GridSearchCV to train the model. Apart from the parameters shown in Table 3, we also search over number of estimators (trees) used in the Random Forest. We found that in every case, the more the number of estimators, the better the accuracy was (which is expected), and hence we fixed it at 25 for our experimental results.

5.3 FFT Features with Neural Network Classifier

For every possible combination of our search space of the meta-hyperparameters like Δ, f and s, we perform a cross-validation over the hyperparameters of the neural network model. The cross-validation is 4-fold and is for optimizing the learning rate (η) and the dropout keep probability (p). The results on set A show a clear trend in the performance of the meta-hyperparameters and we can remove the poorly performing ones from the future searches to save computation time. The results for the neural network are in Table 4.

Table 5 shows the confusion matrices for our best neural network on the two test sets - one of them from individuals the model has never seen while training. The network performs really well, close to 100% accuracy, across all sports when test data is from the distribution of the same individual as the training data. This classifier performs poorly on the test data taken from the individuals it hasn't seen

Table 3: Performance of the Random Forest Classifier on FFT features

Data Sets		Meta-hyperparams (sec)				Optimized hyperparams		Accuracies (%)	
Input	2^{nd} Test	Δ	f	s	comps*	minSamp**	splitCrit***	Test	2^{nd} Test
A	-	15	4	1	100	5	entropy	90.35	-
A	-	15	4	1	200	10	entropy	87.87	-
A	-	15	6	1	100	5	gini	89.36	-
A	-	15	6	1	200	5	entropy	91.34	-
A	-	15	6	1	400	10	entropy	88.61	-
A	-	15	6	2	200	5	entropy	82.18	-
A	-	15	6	2	400	10	entropy	80.94	-
A	-	30	4	1	100	5	entropy	93.53	-
A+B+C	-	15	4	1	100	5	entropy	87.15	-
A+B+C	-	30	4	1	100	5	gini	91.1	-
A	B+C	15	4	1	100	10	entropy	88.61	**42.89**
A	B+C	30	4	1	100	5	gini	93.03	33.25
A+B	C	15	4	1	100	5	gini	92.18	26.36
A+B	C	30	4	1	100	10	gini	**94.62**	17.43

*comps is number of components obtained from PCA.
The units of seconds as mentioned beside meta-hyperparameters title doesn't apply to it.
**minSamp is the minimum number of samples required in node for tree to split on
***splitCrit is the criterion to split the node on: gini or entropy

Table 4: Performance of the Neural Network Classifier with various parameters

Sets		Meta-hyperparams (sec)				Optimized hyperparams		Accuracies (%)		
Input	2^{nd} Test	Δ	f	s	comps*	LearnRate	KeepProb	Train	Test	2^{nd} Test
A	-	15	4	1	100	1.00E+00	0.25	100.00	95.79	-
A	-	15	4	1	200	1.00E+00	0.25	100.00	94.06	-
A	-	15	6	1	100	1.00E-01	0.50	98.33	92.57	-
A	-	15	6	1	200	1.00E+00	0.25	100.00	91.09	-
A	-	15	6	1	400	1.00E-01	0.20	100.00	85.64	-
A	-	15	6	2	200	3.16E+00	0.40	99.69	85.64	-
A	-	15	6	2	400	3.16E-01	0.20	100.00	80.2	-
A	-	30	4	1	100	1.00E+02	0.25	100.00	98.51	-
A+B+C	-	15	4	1	100	1.00E-01	1.00	100.00	89.08	-
A+B+C	-	30	4	1	100	1.00E+01	0.50	98.31	90.39	-
A	B+C	15	4	1	100	1.00E+01	1.00	100.00	94.8	28.06
A	B+C	30	4	1	100	1.00E-02	0.25	100.00	96.02	**36.23**
A+B	C	15	4	1	100	1.00E-01	0.25	100.00	**98.66**	11.36
A+B	C	30	4	1	100	1.00E+00	0.50	100.00	98.46	28.44

*comps is number of components obtained from PCA.
The units of seconds as mentioned beside meta-hyperparameters title doesn't apply to it.

in training, but as we can see from the confusion matrix, it can detect walking and badminton fairly well. There seems to be some relation between skating and walking when sample are taken from multiple individuals. Also, the classifier seems to think basketball looks very similar to badminton and makes mistakes. To remove these inconsistencies, we would need a lot more data from several individuals to derive a more general picture of what a particular activity or sport looks like in the feature space.

6 Discussion and Conclusions

Our results show that FFT features are superior to statistical features for this usecase. Among classifiers, our neural network seems to work better than the random forest classifier.

Coming to the data sets A, B, and C: we see consistently good results for the test set if it's held out from the same distribution as the one from which the training set is drawn. When there is a new test data set, like the 2^{nd} Test set reported in the tables 2, 3 and 4, which is a set that is from people the

Table 5: Confusion matrices for the test set (left) and 2^{nd} test set (right) results using our best neural network

		1	2	3	4	5	6		1	2	3	4	5	6
Badminton	1	**45**	0	0	0	0	0	1	**47**	11	8	2	3	8
Basketball	2	2	**34**	3	0	0	0	2	37	**11**	1	2	1	0
Foosball	3	0	1	**36**	0	0	0	3	6	4	**5**	7	4	47
Running	4	0	0	0	**20**	0	0	4	5	2	0	**5**	0	2
Skating (ice)	5	0	0	0	0	**30**	0	5	2	2	3	6	**2**	69
Walking	6	1	0	1	0	0	**28**	6	5	0	3	14	3	**76**

classifier has never seen training samples for, the classifiers perform relatively poor. This tells us that there is a huge variation in the activity data from one individual to another, albeit reasonably good accuracies turning up for non-sport activities in such cases. If the model is to have good predictive accuracy on this set as well, then we cannot avoid collecting data from several individuals.

But the fact that the predictive accuracy is so good if the individual is the same signifies that our approach can applied to personal fitness trackers - devices which won't be shared among individuals. The tracker can learn from the owner's activity data over short training period - maybe a couple of activity sessions - when the algorithm can be fed the ground truth, and it can learn to detect these activities for this particular individual from then on.

If we had more time for this project, we had numerous things on our minds to try out. First, we would have gotten a much larger data set from more individuals which may have resulted in better robustness in any sport. Secondly, instead of using only two classifiers and comparing their accuracies, we could have implemented more classifiers and used an ensemble of them for better overall accuracies. Third, instead of collecting and analyzing only accelerometer and gyroscope data, we would have explored various other sensors which might have held more revealing data pertaining to any single sport.

A key area of further investigation could be the crafting of new features. We should explore more with the meta-hyperparameters for our FFT-based features, and even come up with more new features. On a related note, it would probably pay off to invest in convolutional neural networks as they can learn to extract features that are helpful in the classification tasks. Just like they detect low level features like edges and corners, or high level features like faces or shapes, in image data, it should be possible to have an analogous behaviour for time-domain and frequency-domain signals.

References

[1] Ravi, N., Dandekar, N., Mysore, P. & Littman, M. L (2005) Activity Recognition from Accelerometer Data. *Innovative Applications of Artificial Intelligence*, pp. 1541–1546.

[2] Bonomi, A. G., Goris, A. H. C., Yin, B. & Westerterp, K. R. (2009) Detection of Type, Duration, and Intensity of Physical Activity Using an Accelerometer. *Medicine Science in Sports Exercise*, 2009 Sep, pp. 1770–1777.

[3] Kwapisz, J. R., Weiss, G. M. & Moore, S. A. (2010) Activity Recognition using Cell Phone Accelerometers. *ACM SIGKDD Explorations Newsletter*, 2010 Dec, pp. 74–82.

[4] Mannini, A., Intille, S.S., Rosenberger, M., Sabatini, A. M., and Haskell, W. (2013) Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine Science in Sports Exercise*, 2013 Nov, pp. 2193–2203.

[5] Zheng, Y., Wong, W., Guan, X. & Trost, S. (2013) Physical Activity Recognition from Accelerometer Data Using a Multi-Scale Ensemble Method. *Innovative Applications of Artificial Intelligence*, pp. 1575–1581.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp. 2825–2830

[7] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015) *TensorFlow: Large-scale machine learning on heterogeneous systems*.

[8] *Sensor Data Logger* - open-source Android Wear app, available on GitHub at `https://github.com/Steppschuh/Sensor-Data-Logger`. Also available on Play Store for direct installation.