Comparing Tree-Based Ensembles and Deep Learning in Human Activity Recognition

Shang Gao* **Frederick Maier*** **Shannon Quinn**[†] **Khaled Rasheed***
Jennifer Gay[‡] **Matthew Buman**[§]

*Institute for Artificial Intelligence, The University of Georgia, Athens, Georgia
†Department of Computer Science, The University of Georgia, Athens, Georgia
‡College of Public Health, The University of Georgia, Athens, Georgia
§School of Nutrition and Health Promotion, Arizona State University, Phoenix, Arizona

Abstract

This study compares the performance of four different classifiers—random forests, extreme gradient boosted trees, long short term memory networks, and convolutional long short term memory networks—on a human activity recognition dataset. The dataset used for this study was collected from 77 subjects performing 23 different activities. Each subject was monitored using a single hip-worn triaxial accelerometer that collected 100 data points per second. Classification accuracy was measured at two different levels of granularity. At the first level, activities were classified as ambulatory or non-ambulatory. At the second level, activities were classified as non-ambulatory, walking, running, upstairs, or downstairs. At both levels, classifiers were tested using both within-subject test/train splitting and between-subject test/train splitting.

1 Background

Human activity recognition is a type of classification problem that involves attaching monitoring devices (e.g. accelerometers, heart rate monitors) to a person while that person is performing different activities. In some cases, external devices such as video cameras may also be employed in the suite of monitoring devices used to collect data. A predictive model is then developed that can identify the activity a person is doing based on the readings from the monitoring devices.

Increasing the number of monitoring devices used to collect data during activities improves the quantity and oftentimes quality of data that can be used to create a predictive model[1]. However, outside of a lab setting, the use of multiple monitoring devices may be both obtrusive and unrealistic. Thus, an ideal solution would create an accurate predictive model using a minimal number of monitoring devices.

Automated human activity recognition has many potential applications. For example, a smart phone app can be designed that captures an individual's activities over a time period and estimate how many calories have been burned[2]. Alternatively, for individuals who work in an office-type environment, an app can remind an individual to take a walk if the individual has been sedentary for too long. These types of apps can also be used to identify individuals who have an active lifestyle for marketing purposes. The increasing popularity of FitBit, an activity tracking mobile app that utilizes a wrist-worn monitoring device, illustrates the growing practical applications of human activity recognition for personal health monitoring[3].

Human activity recognition is challenging because different people may perform the same activity in very different ways. For example, a small lightweight female may have a very different walking

Table 1: Dataset activities description

Activity Description	No of Subjects	Duration/Distance
Treadmill at 1 mph @ 0% grade	29	3 min
Treadmill at 2 mph @ 0% grade	21	3 min
Treadmill at 3 mph @ 0% grade	28	3 min
Treadmill at 3 mph @ 5% grade	29	3 min
Treadmill at 4 mph @ 0% grade	33	3 min
Treadmill at 5 mph @ 0% grade	21	3 min
Treadmill at 6 mph @ 0% grade	34	3 min
Treadmill at 6 mph @ 5% grade	26	3 min
Seated, folding/stacking laundry	74	3 min
Standing/Fidgeting with hands while talking	77	3 min
1 minute brushing teeth + 1 minute brushing hair	77	2 min
Driving a car	71	-
Hard surface walking w/ sneakers	76	400m
Hard surface walking w/ sneakers hand in front pocket	33	100m
Hard surface walking w/ sneakers while carry 8 lb. object	30	100m
Hard surface walking w/ sneakers holding cell phone	24	100m
Hard surface walking w/ sneakers holding filled coffee cup	26	100m
Carpet walking w/ high heels or dress shoes	70	100m
Grass walking barefoot	20	134m
Uneven dirt walking w/ sneakers	23	107m
Up hill 5% grade w high heels or dress shoes	27	58.5m x 2 times
Down hill 5% grade w high heels or dress shoes	26	58.5m x 2 times
Walking up stairs (5 floors)	77	5 floors x 2 times
Walking down stairs (5 floors)	77	5 floors x 2 times

pattern from that of a larger, heavier male. An effective classifier must be able to accurately identify different activities across a wide range of people.

2 Dataset

The dataset for this project consists of hip-worn accelerometer data measurements across 77 subjects performing various activities. An ActiGraph GT3X+ triaxial accelerometer sampling at 100 hertz was used to collect the data. For each subject, the accelerometer was placed on the anterior axillary line of the nondominant hip. Data was collected from 77 subjects between 18-64 years old. All subjects were recruited from the Phoenix, AZ and surrounding areas through community sources, email distribution lists, and social media outlets. All data was collected at Arizona State University in 2013 and 2014.

All participants completed the following activities: seated, folding/stacking laundry, standing/fidgeting with hands while talking, 1 min of brushing teeth and 1 min brushing hair, hard surface walking, carpet walking, and walking up and down stairs. An additional three treadmill activities and three other activities were randomly assigned. Time stamps for each activity were captured using a custom-built Android application that was synced to the accelerometer. The full list of activities is displayed in Table 1.

The accelerometer raw data is approximately 17 million rows and 7 columns (rowid, subject id, activity id, timestamp, x-accel, y-accel, z-accel). In addition, there is demographic data for each subject (77 rows, 8 columns–id, age, height, sex, weight, bmi, weight group, overall group). This dataset is fairly large compared to other datasets used for human activity recognition (many other studies use fewer subjects and fewer activities[4]).

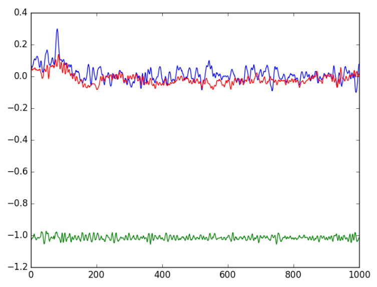

Figure 1: Example accelerometer reading from non-ambulatory activity.

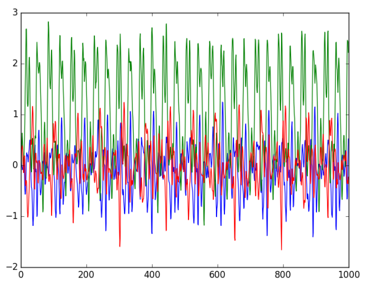

Figure 2: Example accelerometer reading from running activity.

3 Feature Engineering

Many of the classifiers used in this study are unable to take in the raw accelerometer data directly as input features. Therefore, we had to hand-engineer features from the raw data to feed into our classifiers.

The primary method used for feature engineering consisted of generating summary statistics over a time period. This method was chosen due to its effectiveness in previous accelerometer-based human activity recognition tasks[4]. For example, the raw accelerometer data was split into ten-second windows, and summary statistics were then calculated over each window. The following statistics were calculated for the readings on each axis (X,Y,Z), the combined magnitude of all three axis (calculated as $\sqrt{x_i^2 + y_i^2 + z_i^2}$), and the first differential of each axis and magnitude:

- Mean and standard deviation
- 10,25,50,75,90 percentiles
- Min, max, range, interquartile range
- Number of median crossings
- Correlation with other axes

The following normalized demographic data was also included as input features:

- Continuous: age, height, sex, weight, bmi
- Binary: overweight, obese

The summary statistics and demographic data was concatenated to form 109 input features that could be fed directly into a classifier.

Other studies in human activity recognition also use frequency based features derived from Fast Fourier transform or Discrete Wavelet transform[5]. However, adding these features to our models did not improve classification accuracy so frequency based features were not included in our final feature set.

4 Labels and Test/Train Splitting

All classifiers were tested at two different levels of classification granularity. At the first level, activities were classified as ambulatory or non-ambulatory. Non-ambulatory activities included folding laundry, brushing teeth/hair, driving a car, and talking while standing/fidgeting. Ambulatory activities included all other activities.

At the second level, activities were classified as non-ambulatory, walking, running, upstairs, or downstairs. Note that running was defined as any treadmill activity that was 5 mph or faster. We expected all classifiers to perform better on 2-category classification and worse on 5-category classification.

Table 2: Random forest test set accuracy

	2 classes	5 classes
Within-subject	98.16	93.67
Between-subject	96.90	83.74

Furthermore, all classifiers were tested using both within-subject test-train splitting and between-subject test/train splitting. In within-subject test/train splitting, the training data included random samples of data from all 77 subjects. Therefore, classifiers were able to train on data from every subject that would appear in the test set.

In between-subject data, data from 1/10 of the subjects were withheld as a test set. Classifiers were trained on data from the remaining 9/10 of the subjects, and all data in the test set was from unseen subjects. We expected all classifiers to perform better in within-subject test/train splitting than between-subject test/train splitting.

5 Models

In our study we tested the performance of four different classifiers, two of which are tree-ensembles and two of which are deep learning. All models were tested on both 2-category classification and 5-category classification. In addition, all models were test using both within-subject test/train splitting and between-subject test/train splitting. This yields a total of four test accuracies per model:

- Within-subject 2-category classification
- Within-subject 5-category classification
- Between-subject 2-category classification
- Between-subject 5-category classification

For all models, each training or testing instance consisted of activity data over a 10-second time window. However, the feature engineering methodology for each 10-second window differed between the models.

5.1 Random Forests

A random forest is a tree-based ensemble classifier. Random forests utilize multiple decision trees, each of which is fitted on a subset of the total available input features. The prediction of all decision trees is then averaged to create a final prediction. Random forests are resistant to overfitting, train relatively quickly, and have generally good accuracy[6].

For this study, we used the random forest classifier from the Python Scikitlearn package. Our random forest model utilized 100 trees, no max depth, and gini impurity. For each test/train instance, the input features for the model consisted of summary statistics and demographic data calculated over the entire 10-second window. Our model took approximately one minute to train (per fold in K-fold cross validation). The test set accuracies on 10-fold cross validation are displayed in Table 2.

5.2 XGBoost

XGBoost, or extreme gradient boosted trees, is an extension of traditional gradient boosted trees. XGBoost is a tree-based ensemble classifier that starts with a single decision tree and then sequentially adds additional decision trees one at a time. Each subsequent decision tree fits to the errors of all previous combined trees. Unlike traditional gradient boosted trees, XGBoost also incorporates L1 and/or L2 regularization to prevent model overfitting. The final output is the cumulative sum of the output of all decision trees in the ensemble[7]. XGBoost has a reputation for being an extremely effective classifier and was used in many of the winning models for Kaggle data science competitions.

For this study, we used the Python xgboost package. Our xgboost model utilized 300 trees, a max depth of 5, a learning rate of 0.1, and L2 regularization. For each test/train instance, the input

Table 3: XGBoost test set accuracy

	2 classes	5 classes
Within-subject	98.71	95.68
Between-subject	97.35	88.65

Table 4: LSTM test set accuracy

	2 classes	5 classes
Within-subject	98.32	94.83
Between-subject	96.29	80.11

features for the model consisted of summary statistics and demographic data calculated over the entire 10-second window. Our model took approximately two to three minutes to train (per fold in K-fold cross validation). The test set accuracies on 10-fold cross validation are displayed in Table 3.

5.3 Long Short Term Memory Networks

Recurrent neural networks are a type of neural network that allows data to propogate through time, and it is therefore appropriate for time series data. In a regular feedforward neural network, each neuron takes in an input to produce an output. In a recurrent neural network, each neuron takes in not only the input from the current timestep, but also the output from the previous timestep to produce an output. This allows data from previous timesteps to also affect the output at the current timestep[8].

A simple recurrent neural network takes in only the input at the current timestep (t) and the output from the previous timestep (t-1). Because the output at the previous timestep (t-1) takes into account the output from the timestep before it (t-2), and so on to infinity, every timestep in a recurrent neural network theoretically takes into account the data from every timestep that occurred before it. However, in practice, signals in a basic recurrent neural network may be lost after several timesteps[8]. In other words, a basic recurrent neural network may fail to capture the causal effect of signals separated over a long period of time.

The long short term memory recurrent neural network, or LSTM network, addresses this shortcoming by introducing memory cells that can store data over a large number of timesteps and release that data whenever it is needed. An LSTM network utilizes a series of gates that control when data is written into a memory cell and when data is released from a memory cell based on the input at the current timestep. If the input at the current timestep is irrelevant to the data stored in a memory cell, the internal state of a memory cell will be preserved until it is needed[8].

For within-subject test/train splitting, we constructed an LSTM network in Theano with the following architecture:

- Layer 1: 128 LSTM memory cells
- Layer 2: 128 LSTM memory cells
- Layer 3: Softmax
- Orthogonal weight initialization
- ADAM gradient descent with learning rate 0.0002

For each test/train instance, we split the 10-second window into twenty 0.5-second windows, then calculated the summary statistics (with concatenated demographic data) over each 0.5-second window. Therefore, the input features for each test/train instance was composed of twenty separate sets of summary statistics (with concatenated demographic data) representing change over time and had matrix of shape [20,109]. We ran the model on a single NVIDIA GeForce GTX 970 GPU for 1 million iterations, which took approximately four hours to train. The test set accuracies on within-subject test/train splitting are displayed in Table 4.

Table 5: Convolutional LSTM test set accuracy

	2 classes	5 classes
Within-subject	98.13	94.64
Between-subject	95.84	85.42

For between-subject test/train splitting, the above LSTM architecture had problems with overfitting (low train loss and low test accuracy), so we constructed an simpler LSTM network with the following architecture:

- Layer 1: 32 LSTM memory cells
- Layer 2: Softmax
- Orthogonal weight initialization
- ADAM gradient descent with learning rate 0.0002

We use the same input features as the first LSTM model. The test set accuracies on between-subject test/train splitting are displayed in Table 4.

5.4 Convolutional LSTM Networks

Convolutional neural networks are a type of neural network that can automatically generate relevant features from raw data. They are commonly used in image recognition tasks because it can be difficult to manually create features for image data. Instead of using densely connected feedforward neurons, a convolutional neural network uses feature maps composed of neurons with shared weights. These feature maps are used to scan across the input data and learn salient features independent from where they appear in the input data[9]. For example, a feature map used in image recognition may learn to identify what constitutes an eye no matter where that eye may appear in an image.

For our application, we use convolutional layers to scan across our raw accelerometer data and demographic data. The convolutional layers automatically learn relevant features that are useful for activity recognition, so no manual feature engineering is required. The output of the convolutional layers is then fed into an LSTM network as its input features.

For within-subject test/train splitting, we constructed a convolutional LSTM network in Theano with the following architecture:

- Layer 1: Convolutional Layer (64x1x10x10 filter with stride 5x1), ELU activation
- Layer 2-4: Convolutional Layers (64x64x3x1 filter with stride 2x1), ELU activation
- Layer 5-6: 64 LSTM memory cells
- Layer 7: Softmax
- Orthogonal weight initialization
- ADAM gradient descent with learning rate 0.0002

For each test/train instance, we used the raw accelerometer data from the 10-second window and concatenated each row with the demographic data associated with the relevant subject. Because convolutional neural networks require a 4D tensor as input (batch size, input feature maps, image height, image width), the input features for each test/train instance was represented by a 4D tensor of shape [1,1,1000,10], where dimension three represented the number of readings over the 10-second window and dimension four represented the number of features per reading (three axis from accelerometer and seven demographic features). We ran the model on a single NVIDIA GeForce GTX 970 GPU for 1 million iterations, which took approximately four hours to train. Note that the convolutional LSTM model did not take noticeably longer to train than the regular LSTM model because the vast majority of computation is carried out in the LSTM portion of the model. The test set accuracies on within-subject test/train splitting are displayed in Table 5.

For between-subject test/train splitting, we once again simplified our architecture to reduce overfitting. Our simplified convolutional LSTM network had the following architecture:

Table 6: Classifier comparison

	Within Subject: 2 classes	Within Subject: 5 classes	Between Subject: 2 classes	Between Subject: 5 classes
Random Forest	98.16	93.67	96.90	83.74
XGBoost	98.71	95.68	97.35	88.65
LSTM	98.32	94.83	96.29	80.11
Conv-LSTM	98.13	94.64	95.84	85.42

- Layer 1: Convolutional Layer (64x1x10x10 filter with stride 5x1), ELU activation
- Layer 2-4: Convolutional Layers (64x64x3x1 filter with stride 2x1), ELU activation
- Layer 5: 32 LSTM memory cells
- Layer 6: Softmax
- Orthogonal weight initialization
- ADAM gradient descent with learning rate 0.0002

We use the same input features as the first convolutional LSTM model. The test set accuracies on between-subject test/train splitting are displayed in Table 5.

6 Result Comparison

XGBoost had the best performance across all categories of evaluation. LSTMs performed well in within-subject test/train splitting, doing only slightly worse than XGBoost, but overfit in between-subject test/train splitting even with simplified models. Convolutional-LSTMs remained competitive with the other models without any need for human feature engineering, and achieved the second-best score in between-subject 5 category classification, which is arguably the most difficult classification in this experiment.

A comparison of the test set accuracies of all four classifiers is available in Table 6 and Figure 3.

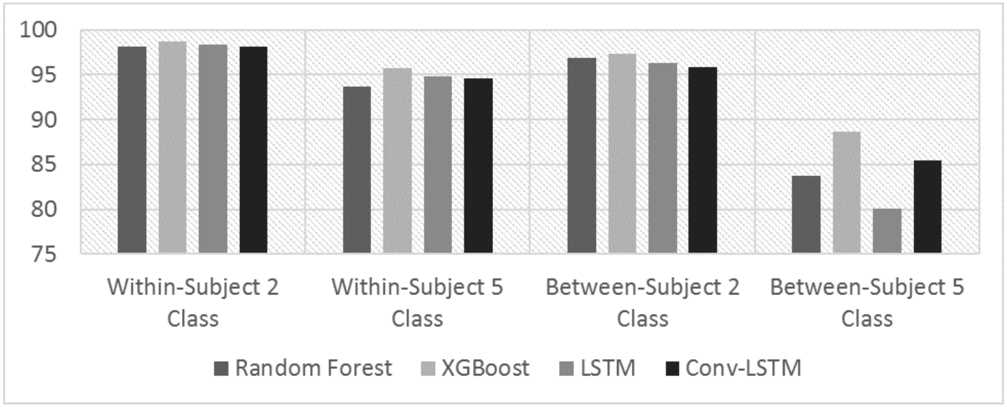

Figure 3: Classifier comparison across different classification categories.

7 Conclusions and Future Work

We conclude that XGBoost is an extremely effective classifier for human activity recognition. Not only did XGBoost achieve the highest test accuracy across all categories of testing, it was also one of the fastest models to train, taking only a few minutes as opposed to the several hours for the deep learning models.

As we expected, 5-category classification is more difficult than 2-category classification because the categories are more granular, and between-subject classification is more difficult than within-subject classification because there is no overlap between the training data and testing data in terms of subjects. In our experiment, the most difficult category was between-subject 5-category classification.

Convolutional LSTMs required no human engineering of features and remained competitive in all categories of classification, and it did comparatively well in the most difficult classification category–between-subject 5-category classification. We believe convolutional LSTMs have much potential in human activity recognition problems and other similar time-series classification problems in which it is not immediately clear to a human the features that may be important.

For future experiments, we suggest running experiments using smartphone-based accelerometers instead of the scientific-grade accelerometers used in this paper. Smartphone-based accelerometers are less precise than scientific-grade accelerometers[10], so it would be useful to determine if these results still hold for smartphone based applications. In addition, data collected in free-living conditions as opposed to a rigorous laboratory setting would better reflect the movement patterns of humans in the real world[11]. Building classifiers for free-living data would be the next step in applying these methodologies to real world applications and may also prove to be a more challenging task.

References

[1] Reiss A., Stricker D. (2011). Introducing a Modular Activity Monitoring System.*Conf Proc IEEE Eng Med Biol Soc 2011*, 5621–5624.

[2] Altini, M., Penders, J., Vullers, R., & Amft, O. (2015, January). Estimating Energy Expenditure Using Body-Worn Accelerometers: A Comparison of Methods, Sensors Number and Positioning. *IEEE Journal of Biomedical and Health Informatics*, 19(1), 219-226.

[3] Shih, P.C., Han, K., Poole, E.S., Rosson, M.B., Carroll, J.M. (2015). Use and Adoption Challenges of Wearable Activity Trackers. *iConference 2015 Proceedings*.

[4] Niazi A., Yazdansepas D., Gay J., Maier F., Ramaswamy L., Rasheed K., Buman M. (December 2016). A Hierarchical Meta-Classifier for Human Activity Recognition, presented at International Conference on Machine Learning and Applications ICMLA'16, Anaheim California, 2016.

[5] Yazdansepas D., Niazi A., Gay J., Maier F., Ramaswamy L., Rasheed K., Buman M. (October 2016). A Multi-Featured Approach for Wearable Sensor-based Human Activity Recognition. *Proceedings of IEEE International Conference on Healthcare Informatics ICHI'16*.

[6] Biau, G. (2012, April). Analysis of a Random Forests Model. *Journal of Machine Learning Research*, 13, 1063-1095.

[7] Chen, T., & Guestrin, C. (2016, March). Xgboost: A scalable tree boosting system. *ArXiv Preprint ArXiv:1603.02754*.

[8] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015, May). A Critical Review of Recurrent Neural Networks for Sequence Learning. *ArXiv Preprint ArXiv:1506.00019*.

[9] Gu, J., Kuen, J., Liu, T., Ma, L., Shahroudy, A., Shuai, B., Wang, Z., Wang, X., & Wang, G. (2015). Recent Advances in Convolutional Neural Networks. *CoRR, abs/1512.07108*.

[10] Sunny, J. T., George, S. M., & Kizhakkethottam, J. J. (2015, September). Applications and Challenges of Human Activity Recognition using Sensors in a Smart Environment. *International Journal for Innovative Research in Science & Technology*, 2(4).

[11] Ellis K., Godbole S., Chen J., Marshall S., Lanckriet G., Kerr J. (2013, November). Physical Activity Recognition in Free-Living From Body-Worn Sensors. *Proceedings of the 4th International SenseCam & Pervasive Imaging Conference*, 88–89.