MSDS-458 Assignment #2

Finding Natural Feature Sets

Mark McGown

❖ Problem Statement

Inferring behavior and actions from phone accelerometer data is an increasingly important area of interest, both to the users who want to know more about their own generalized behavior as well as to companies hoping to cater to that desire. The more accurate these predictions can be made, the more an endless list of possibilities are enabled including: hand-based gestures, patterns of movement when the user has the phone stored, signatures detected by process control systems, medical/fitness tracking, and structural monitoring. These potential applications pose an exciting opportunity for data mining -passive most importantly due to the wealth of data that can be collected while a user is performing other tasks. Accurate predictions in this way may allow interaction with a user's phone that prevents calls during exercise, determine if there's been a car accident to call for help, or simply knowing if a user who has walked upstairs may likely be asleep.

Complications arise with this strategy when two or more actions share similar patterns in accelerometer data. This might occur when such an algorithm is trying to distinguish a user who is going upstairs from a user who is going downstairs. Fortunately, current methods exist such as Long Short-Term Memory (LSTM) that may bring data from these similar actions into focus. Such methods may further grasp the context of the time-based nature for these specific sequences.

❖ Data Set

1.1 million rows of accelerometer data from Fordham University's Wireless Sensor Data Mining (WISDM) lab was used to measure the acceleration of various users performing 6 different tasks in the x, y, and z directions (Kwapisz, Moore, & Weiss, 2010). These directions are shown in Figure A1. Sitting, walking, jogging, standing, going upstairs, and going downstairs were all the actions monitored on 36 different users. One instance of a user sitting is shown in Figure A2 whereas another instance of another user walking is shown in Figure A3. Acceleration is in values reported as 10 = 1 g = 9.8 m/s^s versus time in nanoseconds where 1 sample was taken every 50ms (20 Hz).

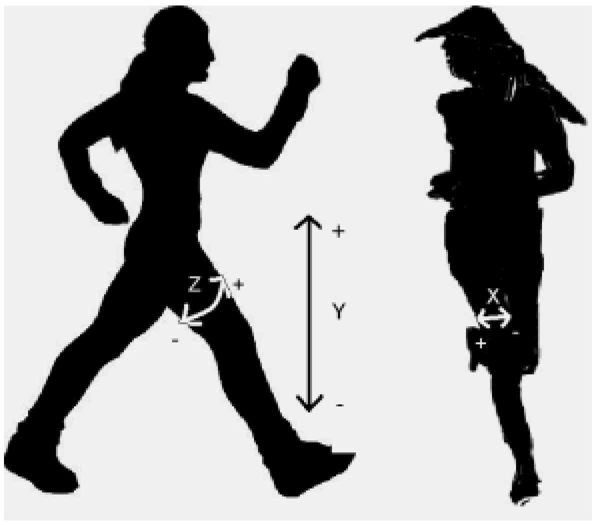


Figure A1: Accelerometer Directions

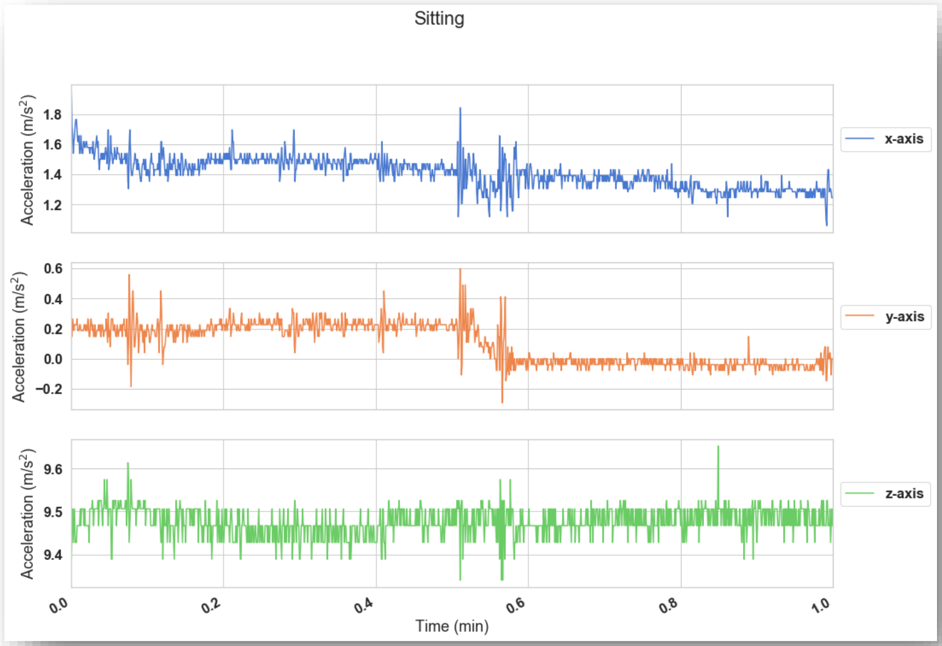


Figure A2: Accelerometer Data of a User Sitting

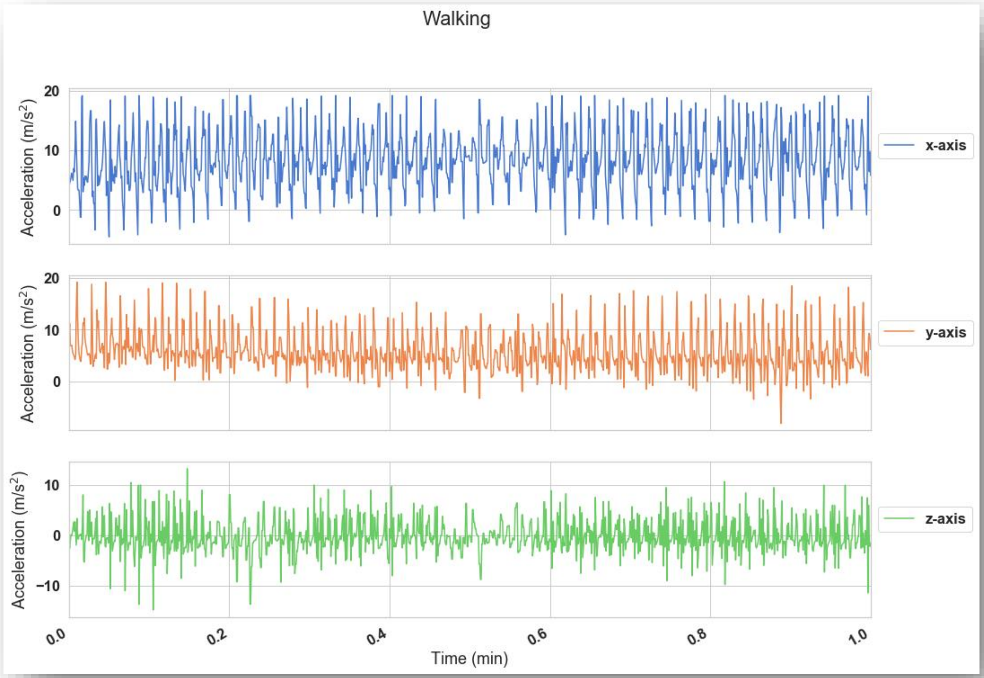


Figure A3: Accelerometer Data of a User Walking

Though the differences between these two actions already may seem potentially stark, Figure A4 & Figure A5 below show the similarities between going downstairs versus upstairs which will pose a more difficult task for certain models. Like walking, they have intermittent regions of higher volatility. For the purposes just of these two illustrations, the time window has been narrowed look more closely at these more active regions to see if more short-term features are visually apparent between the stair actions.

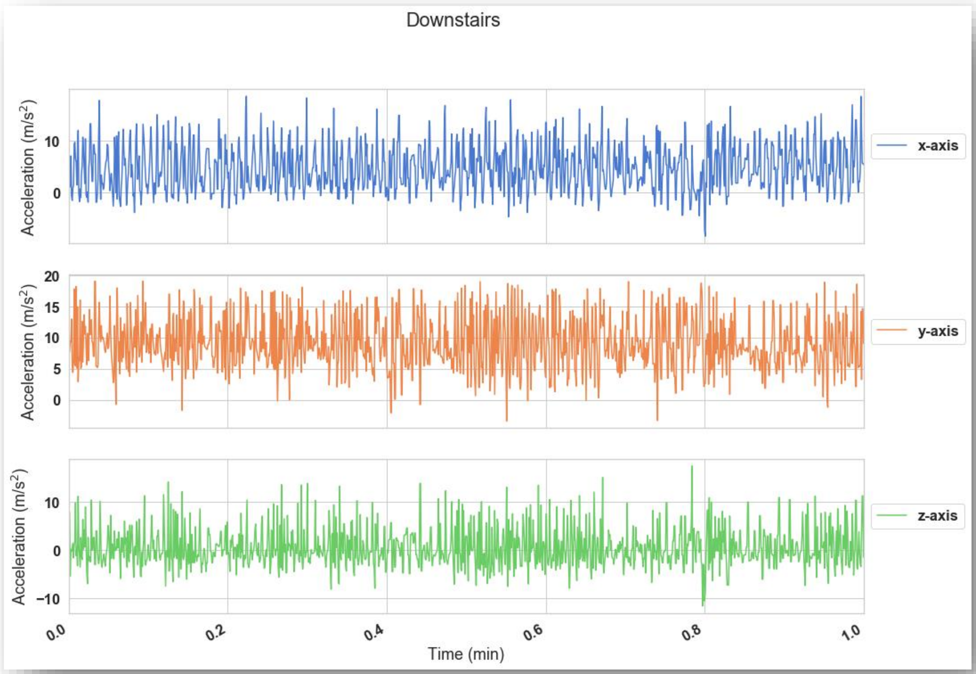


Figure A4: Accelerometer Data of a User Going Downstairs

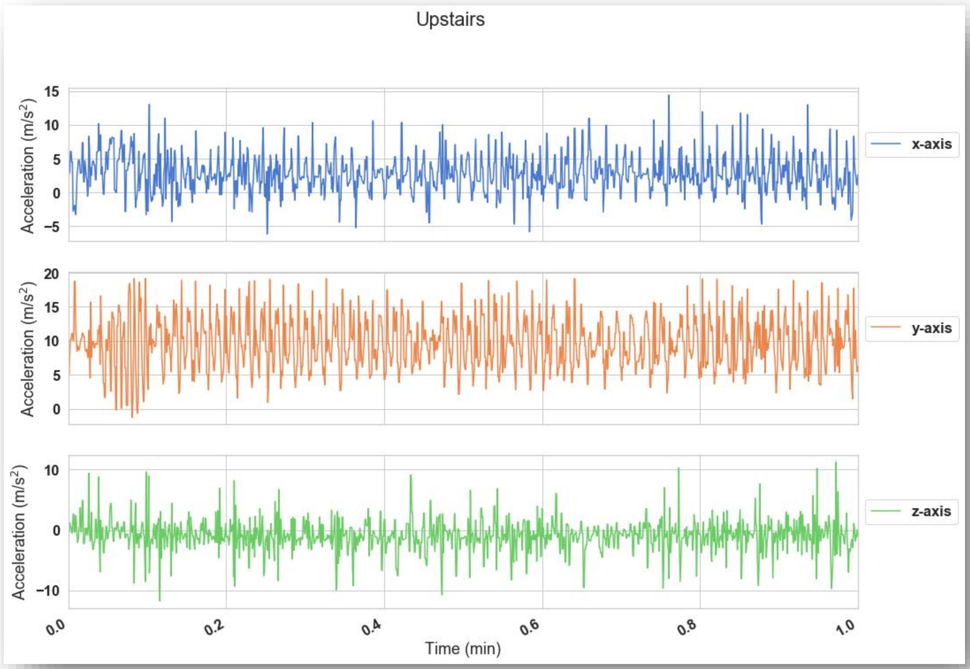


Figure A5: Accelerometer Data of a User Going Upstairs

As one might expect, both actions share similar features. In this case, going upstairs actually looks most like the previous data for walking. Having a dataset with easily discernible features versus those that are more similar will hopefully aid in finding which models have high accuracy across an entire confusion matrix and not just for certain actions in the data set.

The overall number of rows for each action is not equal as shown in Figure A6. One concern might be the underrepresentation of the stairs action classifications. Overall model accuracy/loss for a model on the data set will not be penalized equally, so performance will have to be transparently presented on a per action bases. Figure A7 also shows a grouping of the first four users where not every action was done by every user for the same length of time, either, since the count of rows of data differ within each axis.

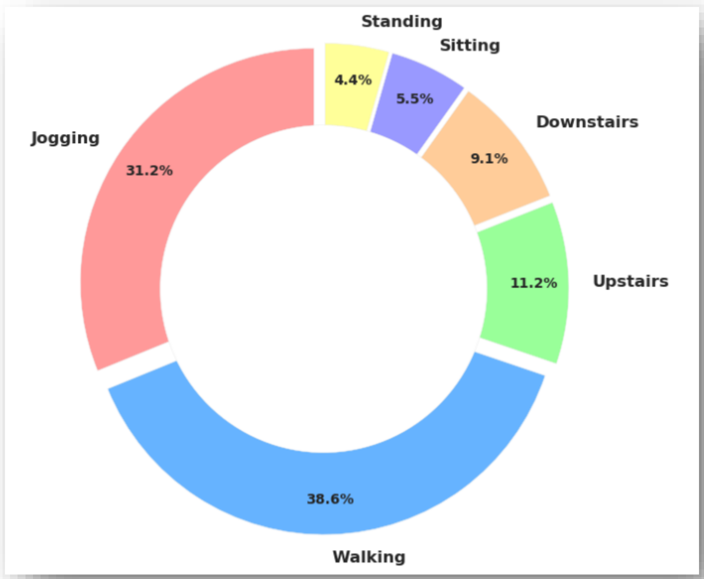
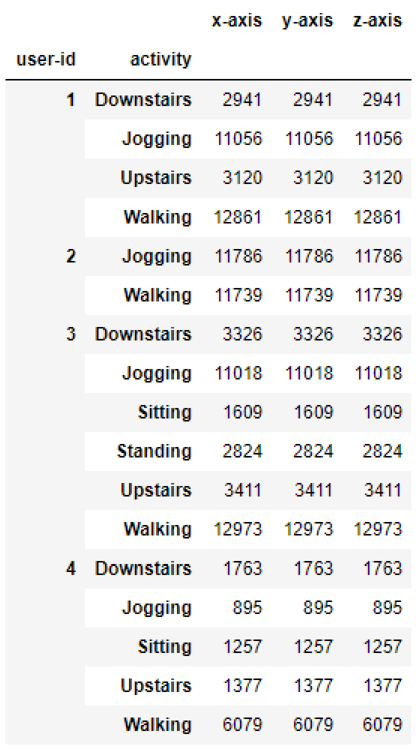


Figure A6: Action Percentages in Data Set

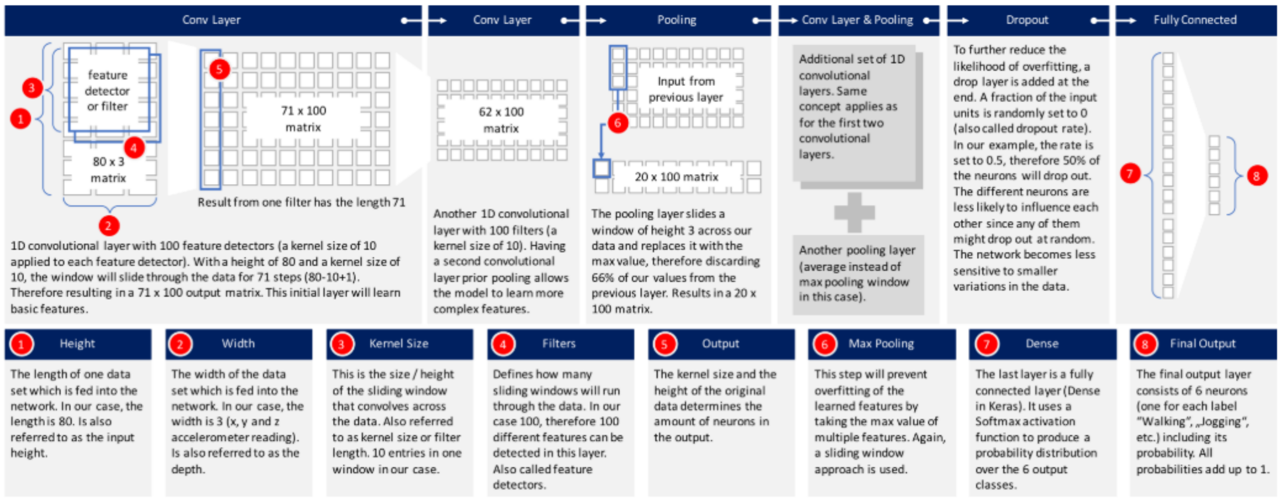


user-id	activity	x-axis	y-axis	z-axis
1	Downstairs	2941	2941	2941
	Jogging	11056	11056	11056
	Upstairs	3120	3120	3120
	Walking	12861	12861	12861
2	Jogging	11786	11786	11786
	Walking	11739	11739	11739
3	Downstairs	3326	3326	3326
	Jogging	11018	11018	11018
	Sitting	1609	1609	1609
	Standing	2824	2824	2824
	Upstairs	3411	3411	3411
	Walking	12973	12973	12973
4	Downstairs	1763	1763	1763
	Jogging	895	895	895
	Sitting	1257	1257	1257
	Upstairs	1377	1377	1377
	Walking	6079	6079	6079

Figure A7: Record Count Snippet by User/Action

❖ Methodology

Two types of temporal NN architecture will be utilized for the final project: Convolutional Neural Networks (CNN) & Long Short-Term Memory (LSTM). 1-Dimensional CNN's and LSTM are chosen due to their noted excellent performance on temporal datasets such as this accelerometer data. Due to its success with this data set, the CNN architecture shown will serve as the foundation of this experiment (Ackerman, 2018). The purpose of each layer used in this foundation is summarized in Figure B.



Result from one filter has the length 71

1D convolutional layer with 100 feature detectors (a kernel size of 10 applied to each feature detector). With a height of 80 and a kernel size of 10, the window will slide through the data for 71 steps (80-10+1). Therefore resulting in a 71 x 100 output matrix. This initial layer will learn basic features.

Another 1D convolutional layer with 100 filters (a kernel size of 10). Having a second convolutional layer prior pooling allows the model to learn more complex features.

The pooling layer slides a window of height 3 across our data and replaces it with the max value, therefore discarding 66% of our values from the previous layer. Results in a 20 x 100 matrix.

Additional set of 1D convolutional layers. Same concept applies as for the first two convolutional layers.

Another pooling layer (average instead of max pooling window in this case).

To further reduce the likelihood of overfitting, a drop layer is added at the end. A fraction of the input units is randomly set to 0 (also called dropout rate). In our example, the rate is set to 0.5, therefore 50% of the neurons will drop out. The different neurons are less likely to influence each other since any of them might drop out at random. The network becomes less sensitive to smaller variations in the data.

1 Height: The length of one data set which is fed into the network. In our case, the length is 80. Is also referred to as the input height.

2 Width: The width of the data set which is fed into the network. In our case, the width is 3 (x, y and z accelerometer reading). Is also referred to as the depth.

3 Kernel Size: This is the size / height of the sliding window that convolves across the data. Also referred to as kernel size or filter length. 10 entries in one window in our case.

4 Filters: Defines how many sliding windows will run through the data. In our case 100, therefore 100 different features can be detected in this layer. Also called feature detectors.

5 Output: The kernel size and the height of the original data determines the amount of neurons in the output.

6 Max Pooling: This step will prevent overfitting of the learned features by taking the max value of multiple features. Again, a sliding window approach is used.

7 Dense: The last layer is a fully connected layer (Dense in Keras). It uses a Softmax activation function to produce a probability distribution over the 6 output classes.

8 Final Output: The final output layer consists of 6 neurons (one for each label "Walking", „Jogging", etc.) including its probability. All probabilities add up to 1.

Figure B: CNN Architectural Layout for Accelerometer Data (Ackerman, 2018)

LSTM's are a special type of Recurrent Neural Network (RNN). RNN's consist of nodes whose activations feedback into themselves and are popularly used to understand the context of a data set. Research some decades ago showed that general RNN's by their very nature had difficulties when the relevant context is not near, in this case recently summarized by the kernel(s) (Bengio, Simard, & Frasconi, 1994). LSTM's are specialized RNN's in that their feedback mechanism has a stronger design against the issue of forgetting relative context, hence their name. Since what a user did before/after going up or down stairs might be relevant to which action they were performing, LSTM's were used over RNN's to include such broader context. This context of each user's actions will be explored using LSTM's to see if important features are more discernable before/after CNN generalization to the overall improvement of the model.

The effectiveness of each non-pooling layer will be explored relative to the resultant accuracy/loss with their removal and/or replacement. Kernel size + filters of the CNN layers, dropout rate, dropout layer ordering, and the addition of LSTM will also be added to this foundation so that the metrics can be compared. Models that achieve higher accuracy in shorter epochs will also be noted. Finally, conclusions will be drawn relative to the nature of the input data set and what lessons can be learned about instructing similar architectural design methodologies going forward.

❖ Preliminary Report on Results

The final dropout layer was first explored when pursuing the most optimal CNN configuration. The goal of the dropout layer is to turn a random set of nodes, at some percentage of the total, to prevent weight adjustment that might overfit the model to noise and/or uncommon features (Ackerman, 2018). The results of these tests are shown in Figure C1 below.

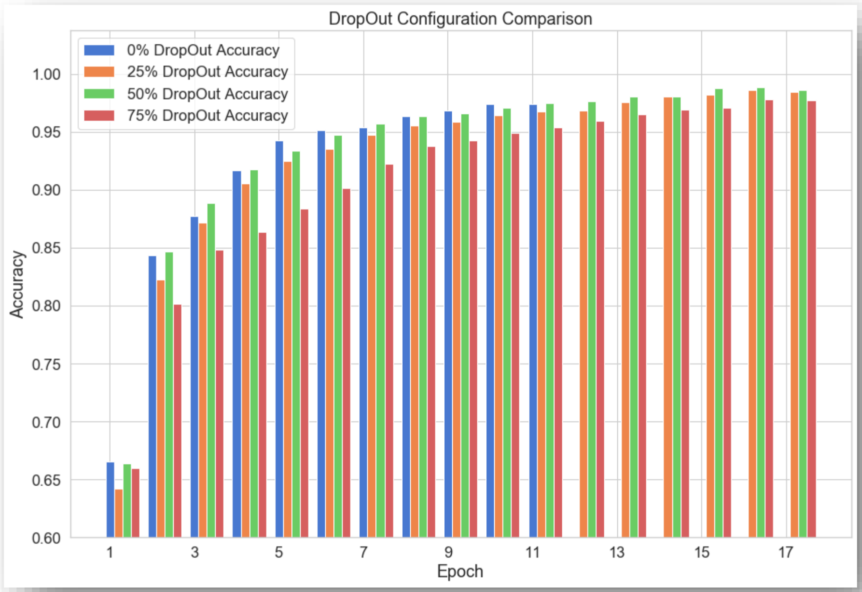


Figure C1: Effect of DropOut CNN Configuration on Model Performance

The termination of the model was set to be when no improvement in accuracy was made for two consecutive runs. Having no dropout layer surprisingly achieved a higher accuracy at earlier epochs, likely since more nodes were able to adjust weights via backpropagation. However, this process terminated model adjustment much earlier than when the dropout layer was present at different dropout rates. Possibly this enabled the model to successfully fine-tune the purpose of each CNN node to achieve higher accuracy in all remaining cases. 50% dropout, the default configuration used by Ackerman, was shown to result in the highest accuracy. A dropout rate of 25% likely made the model still feel the effects of no dropout layer (overfitting) whereas a dropout rate of 75% could have been keeping too many nodes off when they should have been specializing on the correct features over later epochs (underfitting).

Next, two LSTM layers were added right before the dropout layer, once in addition to the 3rd/4th CNN layer and then replacing them entirely. The theory behind this is allowing the model, after the point most generalizations have been made, to maintain memory of the overarching themes behind the different user actions. Figure C2, on the dotted line, shows two LSTM layers of the same size were outperformed by their default CNN counterparts in the solid lines. Excitingly, Figure C2 also shows that keeping all original CNN layers while adding the two LSTM layers found a better accuracy/loss at an even earlier epoch.

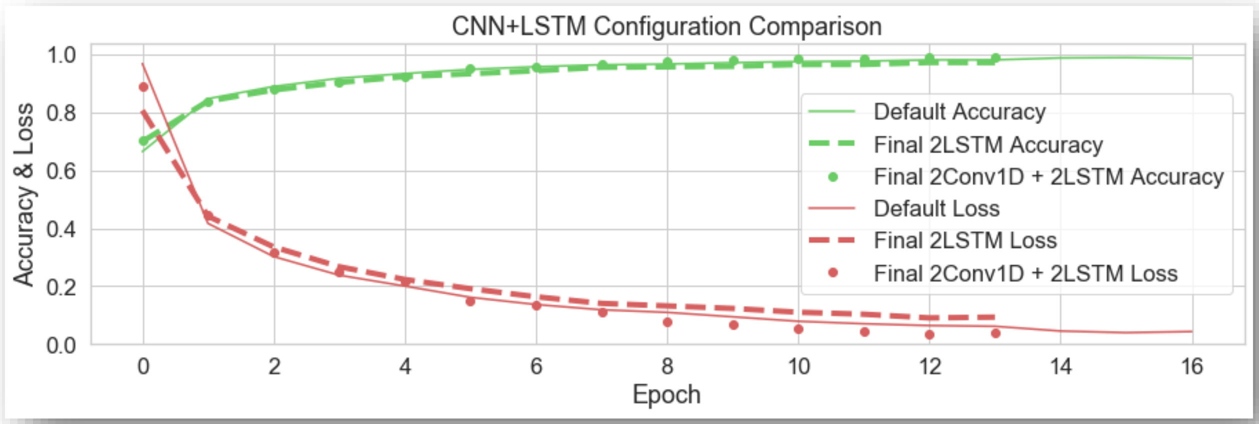


Figure C2: Effect of CNN+LSTM Configuration on Model Performance

❖ Path to Final Project

Moving into the maturity of the exploratory phase, the interplay between generalization and memory will play a strong theme in the conclusions drawn as to what makes a good model for accelerometer data. Overfitting/underfitting of these configurations will also be optimized within their parameters. Dropout layer placement and configuration have proven optimal in their default configurations so far. Most importantly, the default CNN template chosen has both been a highly successful model with a 98% accuracy that has early indications for improvement in other temporal-based methods.

The confusion matrix and the effectiveness of the model on activities under-represented in the data set still need to be explored in-depth. CNN's, with their unique architecture, are already known to excel in temporal data sets including accelerometer data. Likewise, their Conv2D counterparts are also known to excel at image recognition. The goal of the final paper for this project is to try to get closer to a CNN+LSTM configuration that can achieve higher overall accuracy/loss metrics while bringing down the corner-cases of applications thinking a user is going upstairs when they are going downstairs or vice versa.

References

Ackermann, N. (2018, September 14). Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences. Retrieved October 10, 2019, from https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf.

Bengio, Y., Simard, P., & Frasconi, P. (1994, March). Learning long-term dependencies with gradient descent is ... Retrieved October 11, 2019, from http://ai.dinfo.unifi.it/paolo/ps/tnn-94-gradient.pdf.

Kwapisz, J. R., Moore, S. A., & Weiss, G. M. (2010). Activity Recognition using Cell Phone Accelerometers, Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10), Washington DC. http://www.cis.fordham.edu/wisdm/includes/files/sensorKDD-2010.pdf