MSDS-458 Assignment #4

Exploration of CNN & LSTM Architecture:

Optimization & Architecture

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, upstairs, and 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).

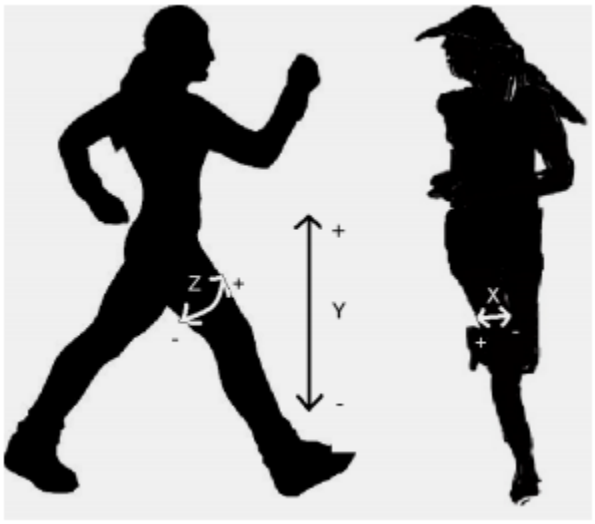

Figure A1: Accelerometer Directions

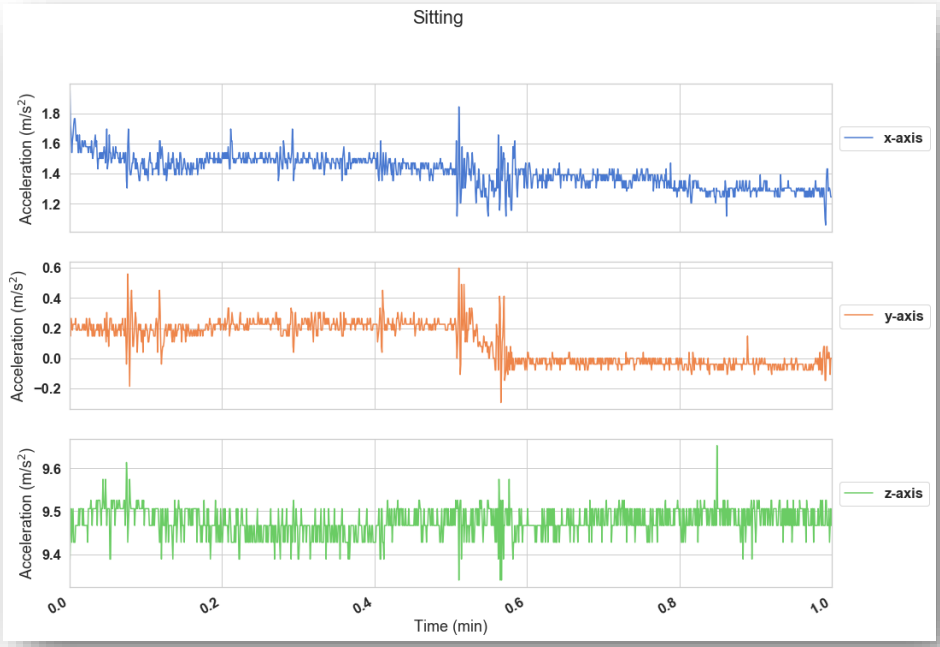

Figure A2: Accelerometer Data of a User Sitting

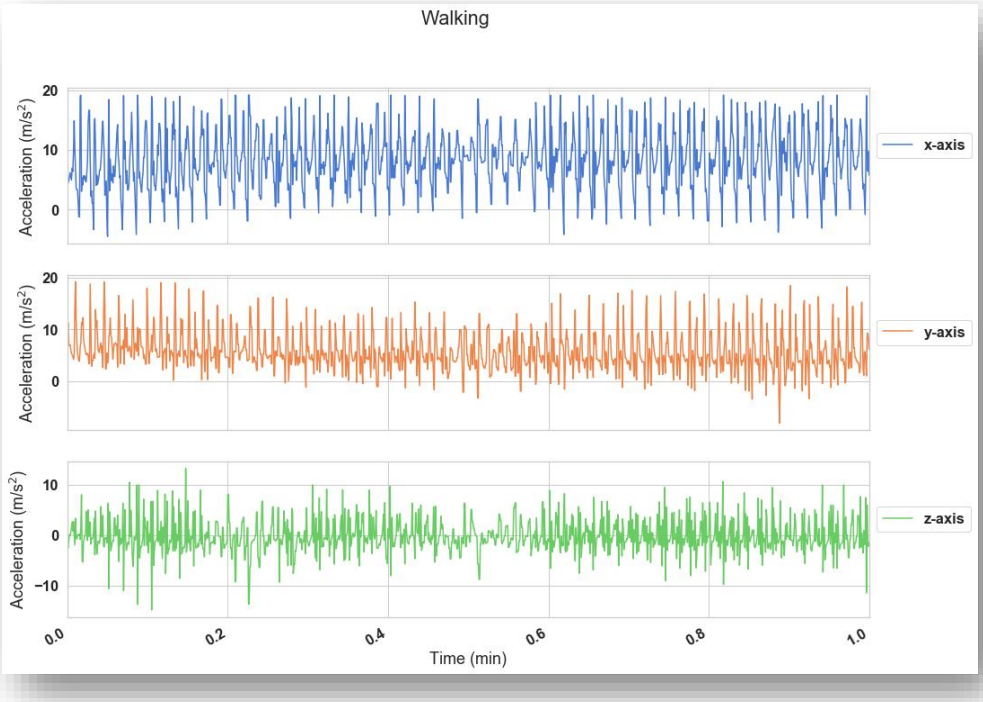

Figure A3: Accelerometer Data of a User Walking

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.

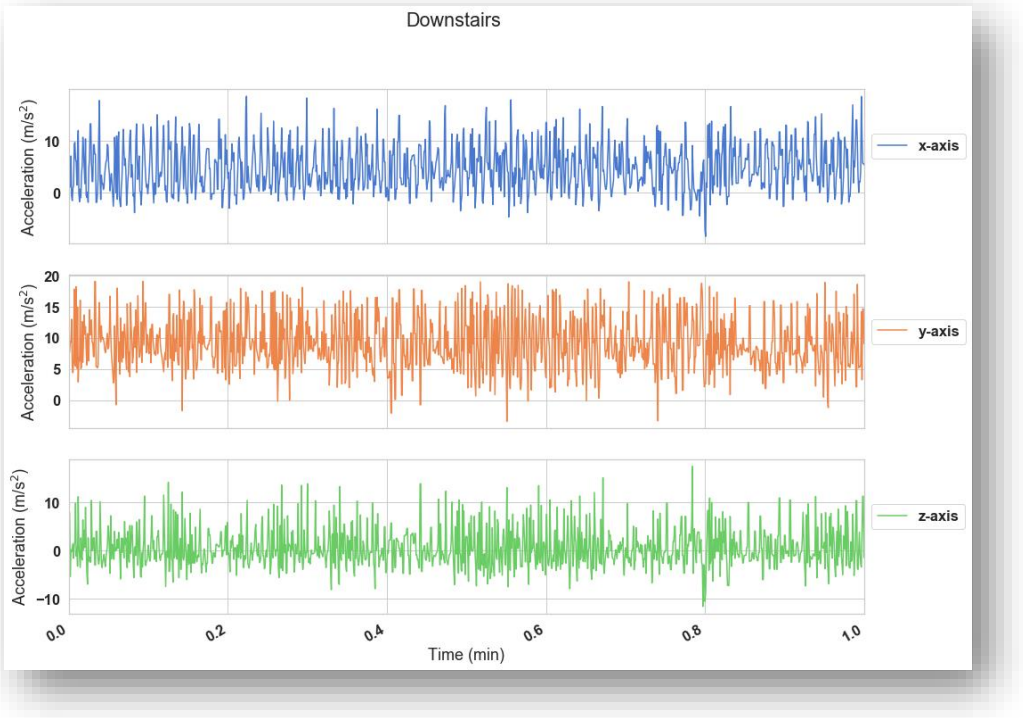

Figure A4: Accelerometer Data of a User Going Downstairs

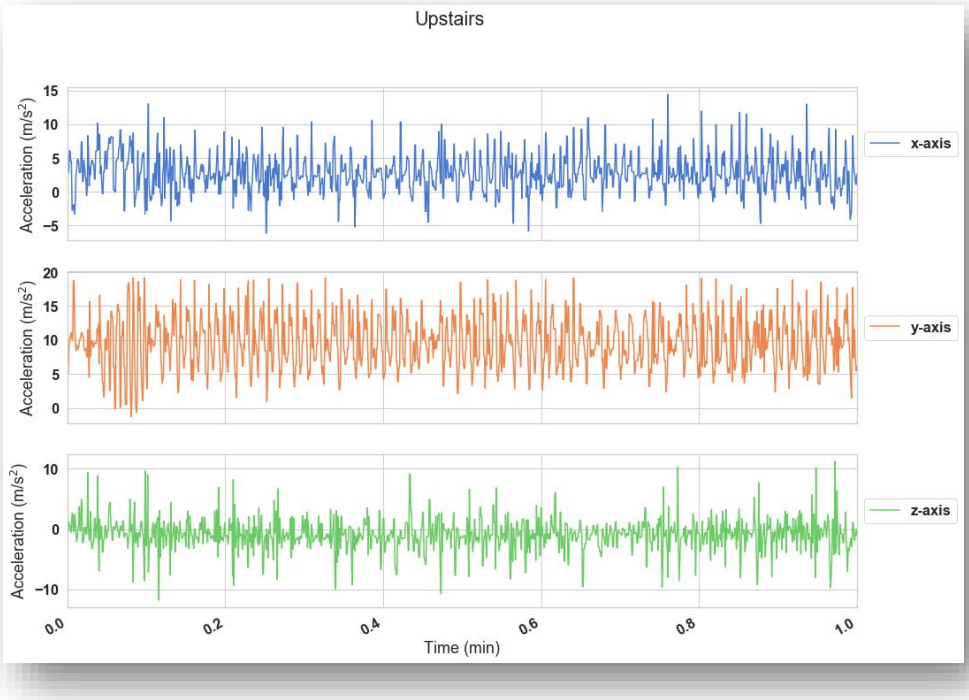

Figure A5: Accelerometer Data of a User Going Upstairs

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 differs within each axis.

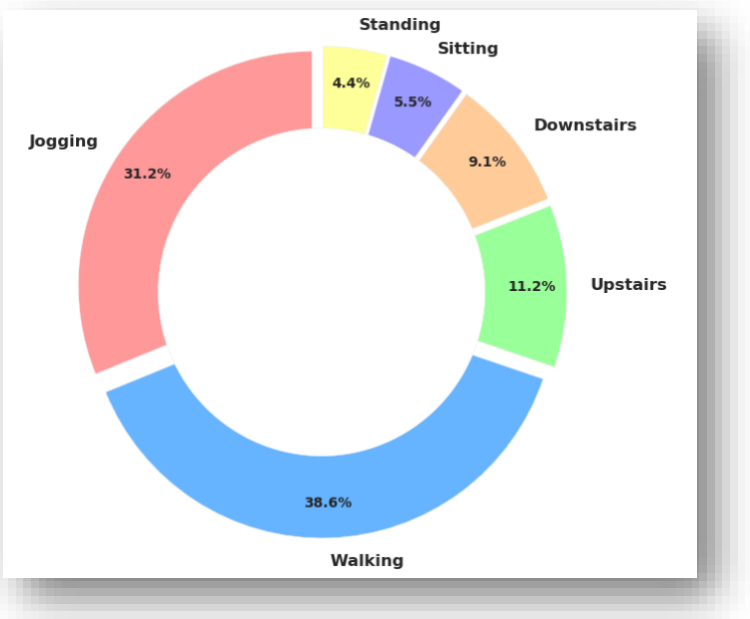

Figure A6: Percentage of Actions (Rows of Data)

		x-axis	y-axis	z-axis
user-id	activity			
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

The CNN's architecture alone was first explored to determine the adjustment of weights over epochs without the influence of LSTM layers in the design. Figure B1 below breaks down the overall layout of this purely CNN design as well as the theoretical purpose of each layer. The optimal parameters of this design, including the removal of certain layers, found the default design provided by Nils Ackerman to perform the best as shown by the training metrics for optimization in the section to follow (Ackerman, 2018).

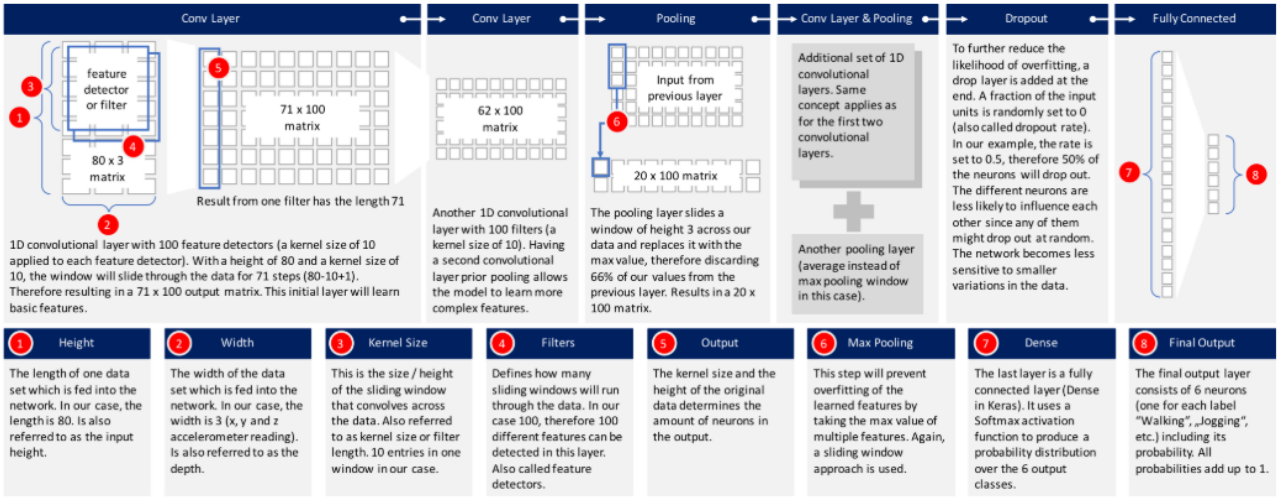

Figure B1: CNN Architectural Layout for Accelerometer Data (Ackerman, 2018)

Rectified Linear Unit (ReLU) was chosen as the activation function (*y = max(0, x)*) for this CNN, which also happens to be the most common activation function for CNN's. This function shown in Figure B2 is interesting considering it is part no activation function in two different senses: if the feature becomes unlearned to this function it may dropout entirely and remain at 0 activation -otherwise the feature has a linear relationship which is the default in Keras for no activation function used (linear regression model). Together, they let undesired features

become unlearned while turning on desired features -all without being computationally expensive.

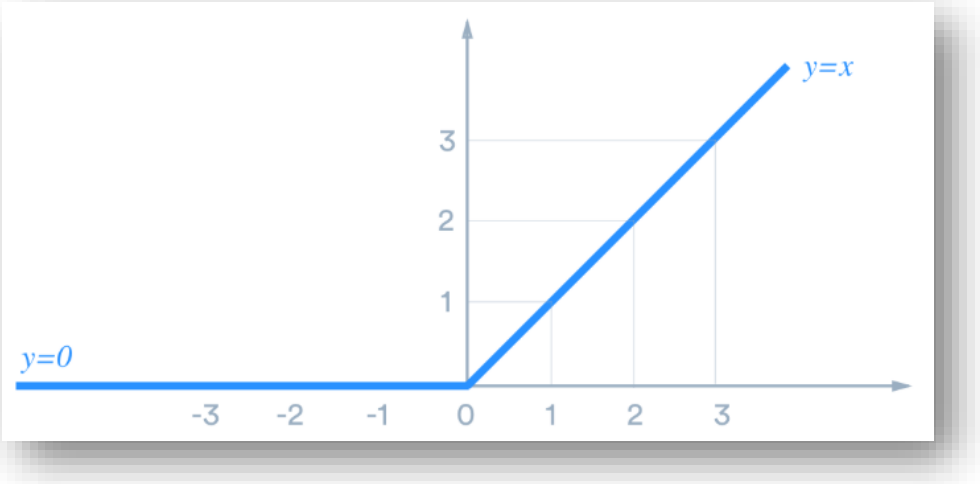

Figure B2: Rectified Linear Unit (ReLU) Activation Function

The softmax function will be used in the last dense layer as the final activation function. This function is necessary for the model to correlate the output of the final DropOut layer with a probabilistic approach to predict which classification, or action, the output corresponds to. The goal by the DropOut layer is to have unique signatures of each feature convolved and cleaned up such that their probabilities binned by the softmax function has as little overlap as possible between classes. The output from the dense layer will partly show which actions are being confused for others, such as stairs actions, but they will also be examined looking at a confusion matrix and model accuracy.

First, the optimization of the parameters that achieve a high degree of accuracy for Nils Ackerman's architecture on this data set will be explored. This will be done by removing certain Conv1D layers, changing their layer size, changing their kernel size, and modifying the DropOut rate that occurs at the end of this CNN architecture. Next, the downstairs action will be traced

through the full CNN model by showing the average output/activation at each layer from input to dense layers. Certain activation signatures will be compared regarding why they were convolved into their final shapes. Finally, some incorporation of LSTM architectures will be explored regarding how they affect the overall convergence of the full layer activations.

❖ CNN Optimization

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.

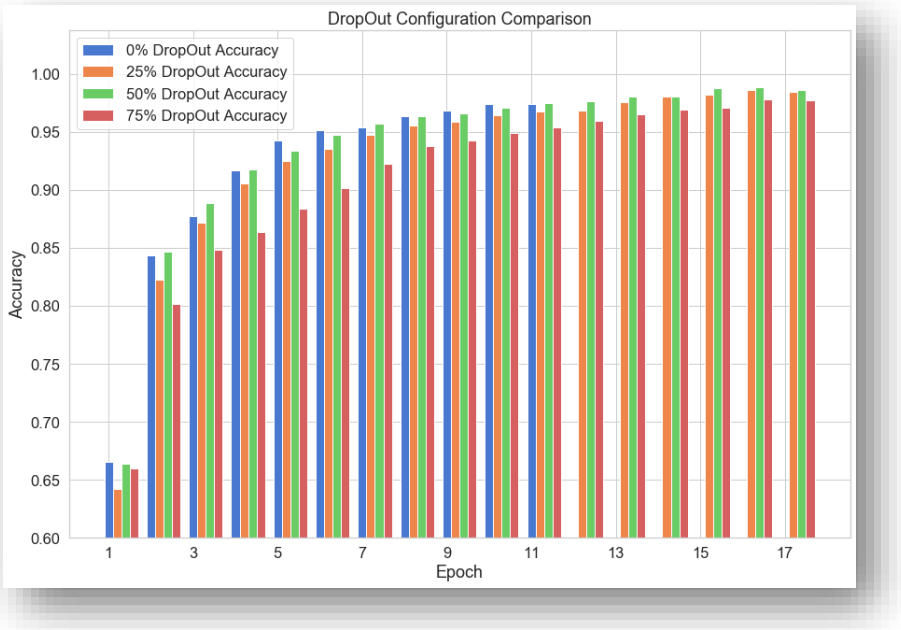

Figure C1: Effect of DropOut CNN Configuration on Model Performance

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). On the other hand, a dropout rate of 75% could have been keeping too many nodes off when they should have been learning the correct features over later epochs (underfitting).

Figure C2 and Figure C3 below show the results of modifying the number of filters and kernels respectively for the early Conv1D layers.

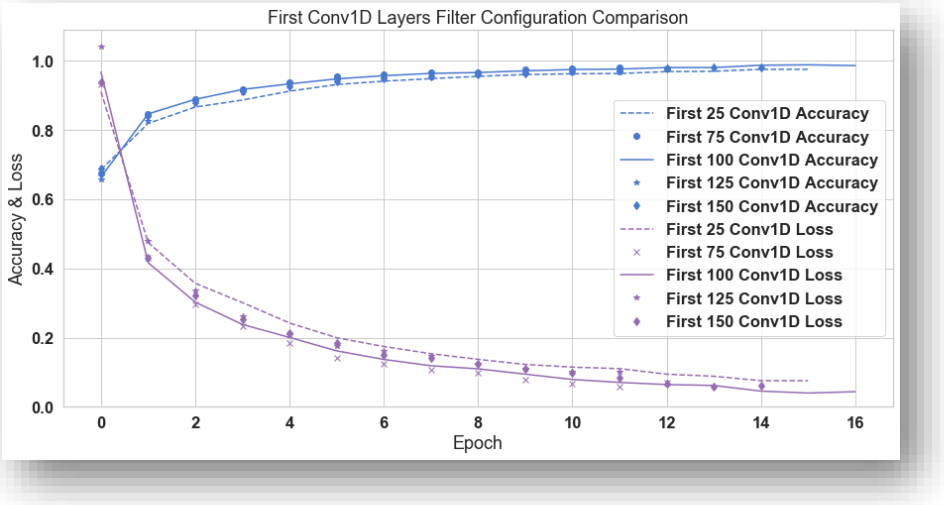

Figure C2: Accuracy Effect of Early Conv1D Filter Count

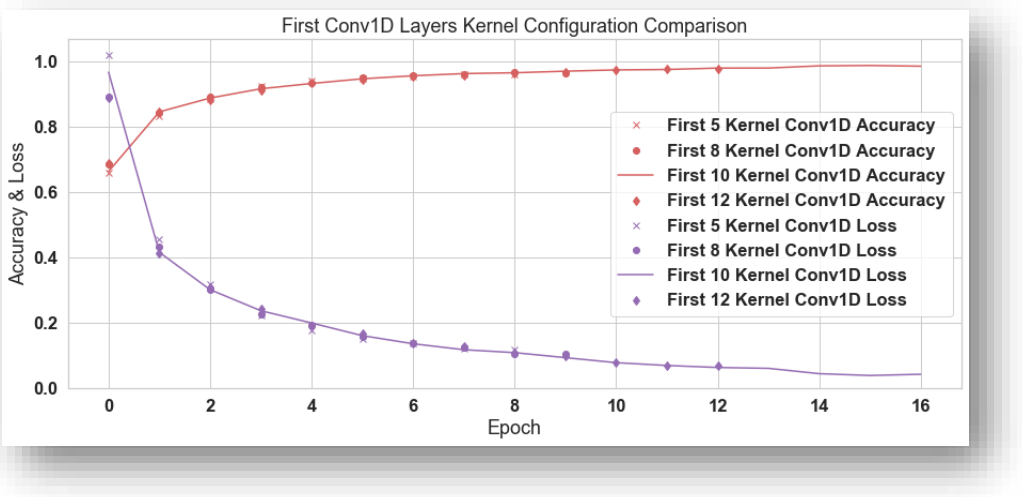

Figure C3: Accuracy Effect of Early Conv1D Kernel Width

The default parameters of 100 filters with 10 kernel width per Conv1D ultimately still proved to be the best for the model on this data set. Interestingly, sub-optimal filter size resulted in lower accuracy achievements even from early epochs whereas sub-optimal kernel widths, like sub-optimal DropOut, merely caused the model to cease for lack of further improvement in later epochs. This may indicate that a kernel width too small or large generally may convolve the general features of the classes similarly to subsequent layers but at a certain point may be too underfit/overfit for the finer tuning the default model performed.

Sub-optimal filter performance had the more dramatic effect of degrading accuracy from early epochs while still taking almost the same number of epochs for convergence. As we will see, the number of filters corresponds to the resolution of the peaks of accelerometer data. Like kernels, optimizing this parameter is important to avoid underfitting/overfitting, except exceptional degradation to layer size adjustment may indicate the significant features were more unique in their accelerometer data height (m/s^2) rather than their width (temporal).

To explore if the default model needed two final Conv1D layers following max pooling, the accuracy over epochs was compared between the default model, the default model minus a Conv1D layer after max pooling, and the default model minus both Conv1D layers after max pooling. These results are shown in Figure C4. Again, the default model achieved the best accuracy in the shortest period, though the model missing one Conv1D layer came close after many more epochs. The model without both Conv1D layers converged at earlier epochs at a far worse accuracy. Interestingly, the model with both later Conv1D layers missing mirrors that of the default model below in that the improvement made by weight adjustment over later epochs was largely the same. Removing the default's two final convolving layers caused the model to underfit the data despite its notably similar improvements over later epochs.

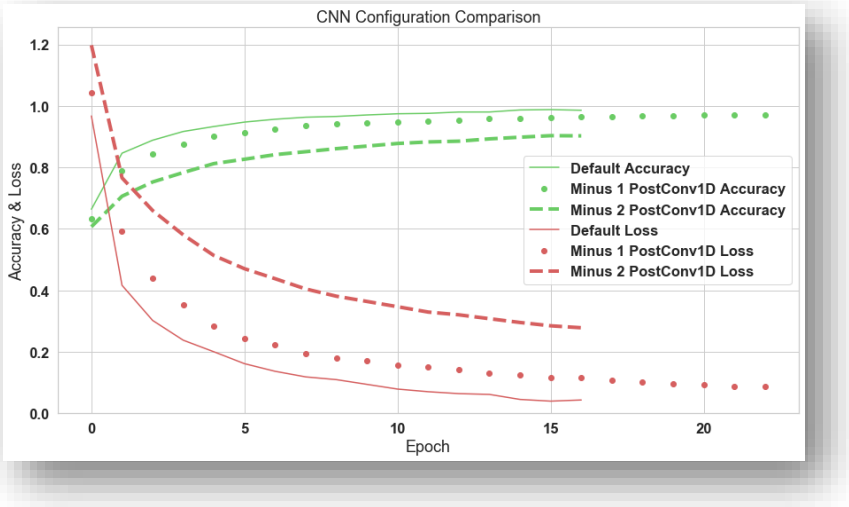

Figure C4: Post Max Pooling Conv1D Comparison

❖ Early Convolutional Layer Function

A full layer heatmap was created to provide further visuals into the inner-workings of each layer in Figure D1 which will be frequently referred to when discussing CNN optimization.

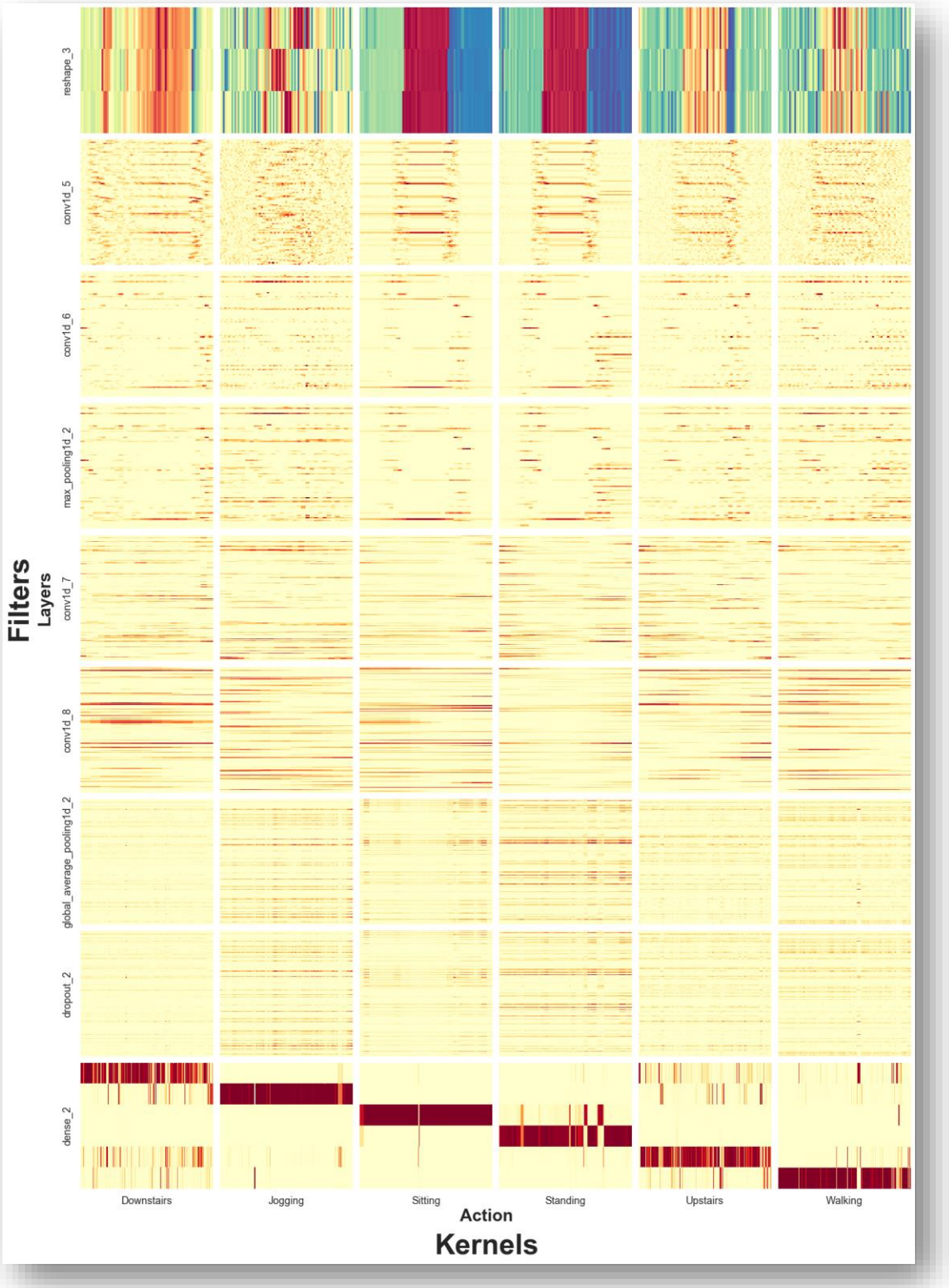

Figure D1: CNN Output & Activation by Layer & Action

Figure D2 is a closer look at the reshaped input layer in the top-left cell of Figure D1, the acceleration of the downstairs action for all three axes over time. This is the same representation of previously shown downstairs raw data (Figure A4) except stacked now to be input into the CNN model. Already, at least three distinct regions emerge of potential interest over all three axes around the same time -each close to the beginning, middle, and end of the time sequence the action was performed. This may represent the user accelerating to go down the first step, building up to a steady accelerated pace as they proceed, and quickly decelerating once they have reached the bottom (in blue).

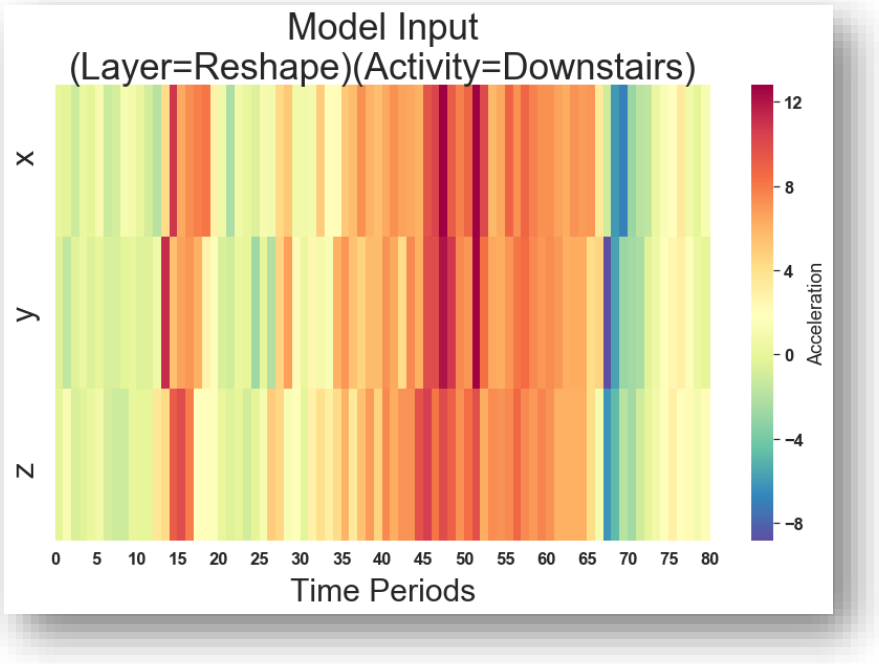

Figure D2: CNN Reshaped Layer for Downstairs Action

Next, we will look at how the first Conv1D layer interprets this data in Figure D3 below. For the y-axis, what was previously acceleration of the different axes has now been broken up into 100 Filters. Kernels with width 10 have convolved the data from a time period of 80

segments at 20Hz (4 seconds). With no padding, each subsequently convolved layer will attempt to summarize the previous layer's features by generalizing the outputs/activations of the layer prior. To make such generalizations about features in a fixed number of kernels, there is a limit to how noisy/complex a data set can be. As we will see throughout this experiment, there must be unique patterns that can be convolved through the noise before the reduced number of kernels available is exhausted.

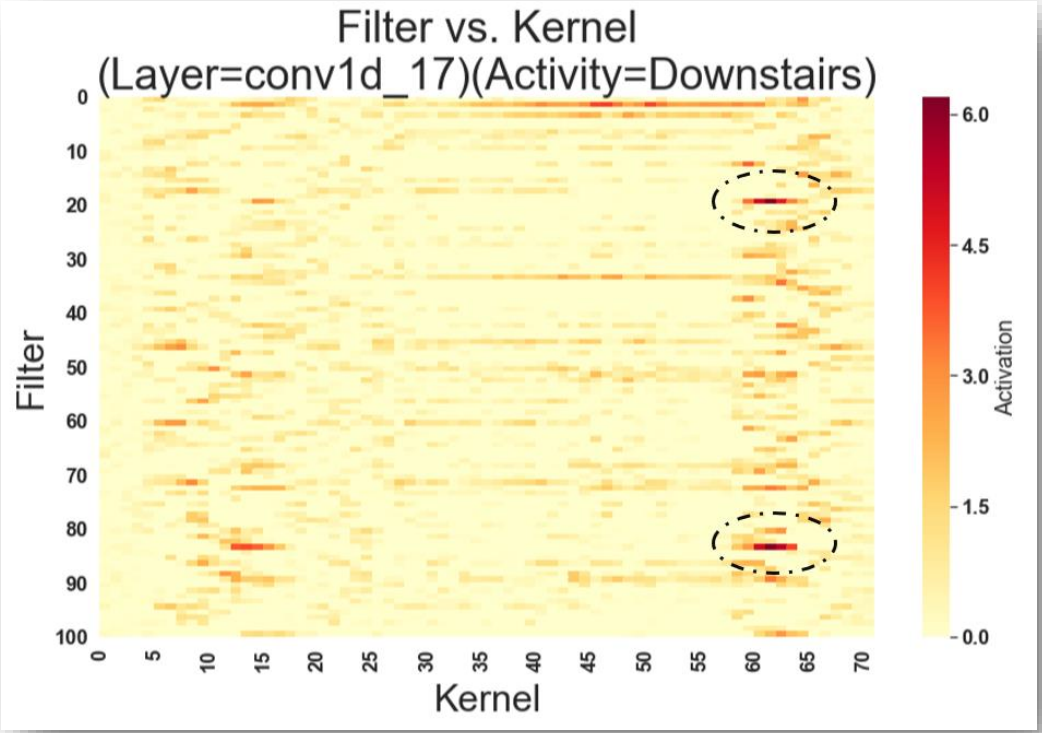

Figure D3: First Conv1D Layer for Downstairs Action

Unlike the input layer for the action of downstairs, only two regions are mainly seen across all layers after their first convolution: those in early and late kernels. In the end, around the 60th kernel, two dark spots of activation can be seen towards filters in the upper and lower

portions of Figure D3 (see arrows) corresponding to action on the x and z axes but noticeably missing the y-axis. In the original input of the downstairs action, a darker red region can be seen on the y-axis before this deceleration, more so than in the x and z axes. This acceleration followed by deceleration has been convolved to already by a weaker feature, possibly in part due to the noisiness of two opposite readings in short succession.

To take a closer look as to what is activating the regions of this action, we will overlay the input reshaped layer, averaged over the axes, with the average activation per kernel across each of the filters from Figure D3. Already we can see the important features being simplified in Figure D4. In the acceleration, we can again see the quick acceleration, gradual acceleration, and the quick deceleration -with the last signature being the most activated region, on average.

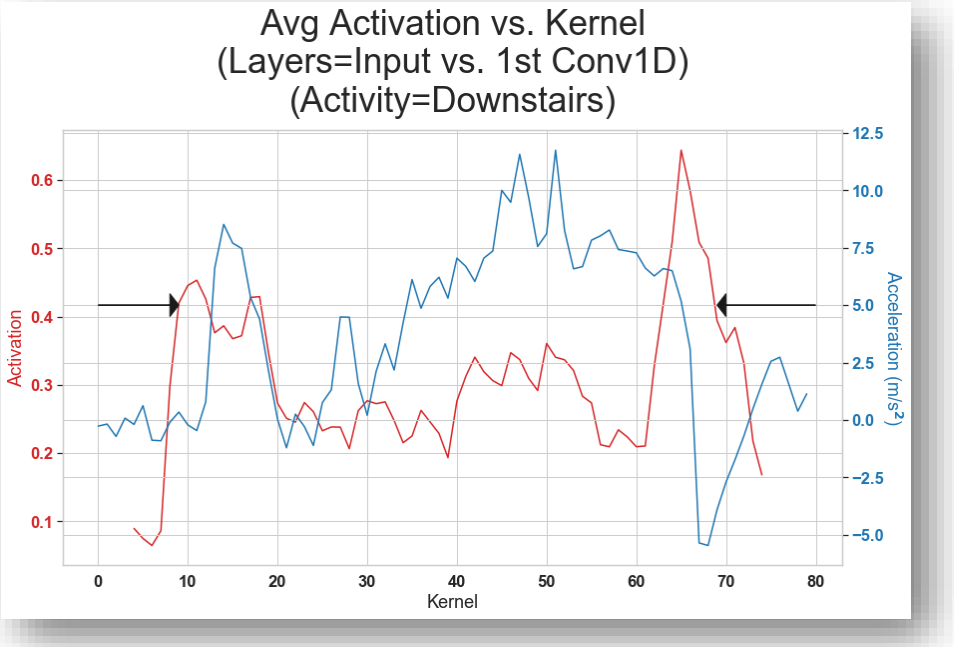

Figure D4: Avg Axis Input vs. First Conv1D Layer Activation

The second Conv1D layer following the first is a further refinement of distinguishing important features from the considerable amount of noise still present in the output of the first Conv1D layer. The third row of heatmaps in Figure D1 shows this noise cleanup for all actions in this experiment. Already, walking/jogging, downstairs/upstairs, & sitting/standing start to share similarities at the second Conv1D layer whereas previously downstairs/jogging could have looked similar due to all the indistinguishable features. A model made just up of two Conv1D layers finds what one would expect: improved accuracy in predictions from just one Conv1D layer, but still a great deal of confusion in prediction between the similar action pairs.

❖ Max Pooling Layer

Pooling differs from convolution in that it is performing a computationally inexpensive operation such as maximum/average over a kernel window. No features are being learned through weighting at the pooling layers; rather, the features learned that already exist are manipulated in some way to make them more learnable. Max Pooling, for example, is commonly used to clean up the edges of blurry numbers/letters during Conv2D training such that the ambiguity of the image is removed and grey pixels next to straight edges are removed. In Figure E below, 2 kernels are iterated over to find the max signal, noticeably retaining the general shape of the original figure more so than the effects of the earlier Conv1D layers. All the major peaks and valleys remain in this image whereas most of the noise/sub-features on those peaks have been removed.

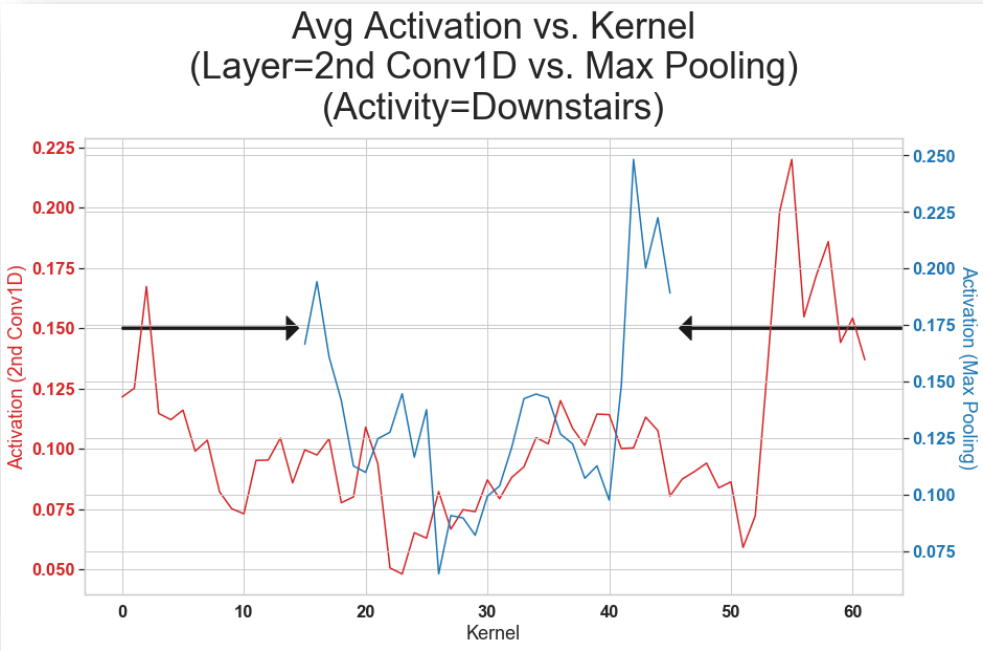

Figure E: 2nd Conv1D vs. Max Pooling Layer Avg Activation

❖ Later Convolutional Layers

Now that the features have been cleaned up, they need to be better learned through the second set of two Conv1D layers. Figure D1 shows this clean-up for all actions in these two layers. The heatmaps displayed are becoming more uniformly red across the activated layers because the kernel size, like before, is reduced after further convolving the weighted features. The change in the features learned in this profile of activation between the first Conv1D layer and now the fourth Conv1D layer can be seen in Figure F1. Like the addition of the second Conv1D layer, the fourth Conv1D was necessary to be added in response to poor accuracy obtained with just three convolutional layers. It is difficult to intuitively know an architecture before observing the challenges in training the model w/r/t noise and other features it may pick up for other classifications e.g. actions, in this case.

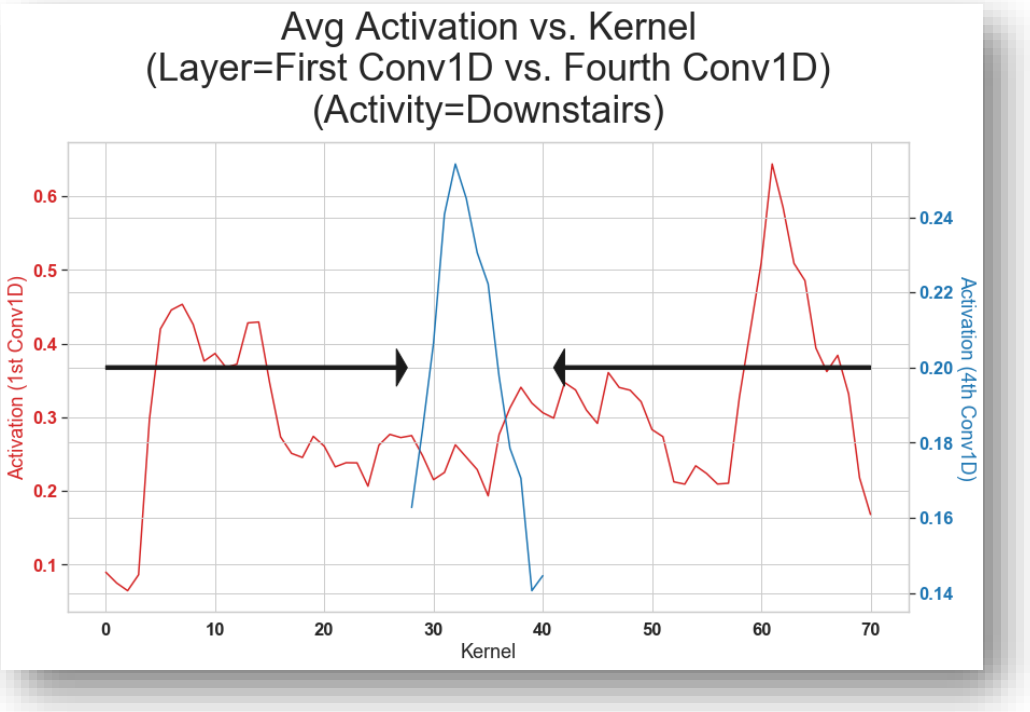

Figure F1: 1st Conv1D vs. 4th Conv1D for Downstairs Action

The feature that emerges in Figure F1 after convolving for the fourth time is the peak of activation first observed temporally at the end of the downstairs action -that is, the activation spike average across all layers that coincided with the rapid deceleration of the user. This can now be compared to the upstairs action signature for this layer as shown in Figure F2 below. One would expect these signatures to appear similar due to the similarity of the action performed and in their similarities in Figure D1. Yet, the signature of the upstairs acceleration through these layers is more apparent as the beginning of a peak rather than the end of one like the downstairs action.

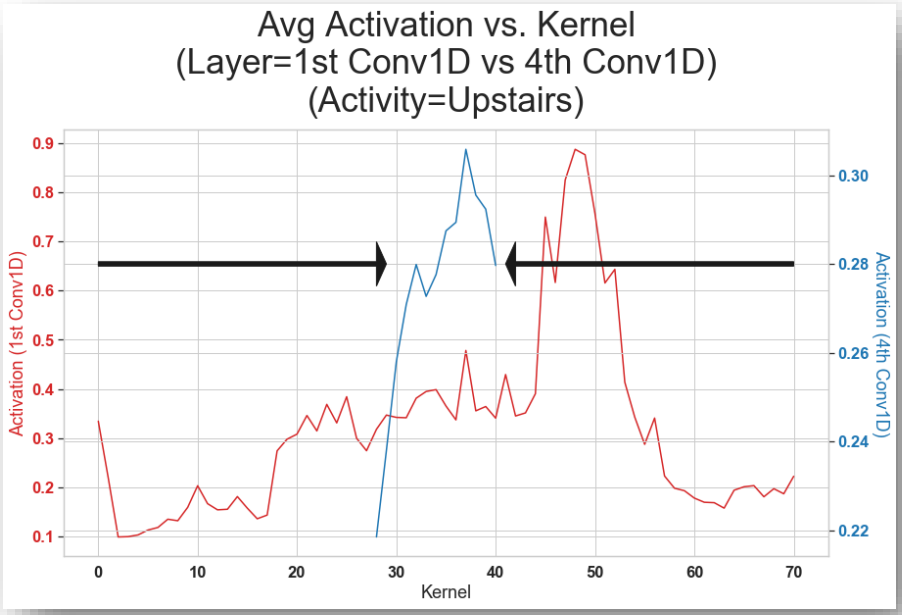

Figure F2: 1st Conv1D vs. 4th Conv1D for Upstairs Action

❖ Global Pooling Layer

Like average pooling, the goal of the global pooling layer is to once again reduce overfitting now that the most significant features have been learned. Now that more noise has been reduced after the first CNN layers, higher accuracy was achieved with global average pooling. This finding is of uniqueness to this architecture relative to this data set. Typically, max pooling is much more common in CNN's so the question for further research is why average pooling was more effective in this instance. One possibility, comparing Figure F1 to Figure F2, is that short regions of activation like the shoulder of the upstairs peak could be wiped out by max-pooling when average pooling would more often take it into account. Further examining temporal 1D data with short time windows, similar behavior amongst classification categories,

and substantial noise would prove interesting to see if global average pooling was part of the optimal model.

❖ CNN Results

The best overall accuracy of the CNN model on the test data was 86%. The confusion matrix for the test classification is shown in Figure G below. As suspected, a great deal of downstairs action was classified as upstairs, though not vice versa. Overall, the model works very well especially on dissimilar actions -which, when considering phone applications today, typically are employed the most often perhaps to avoid this type of confusion. As it will be touched on following the LSTM discussion, a great deal of other phone-based data can be used to drive misclassification down.

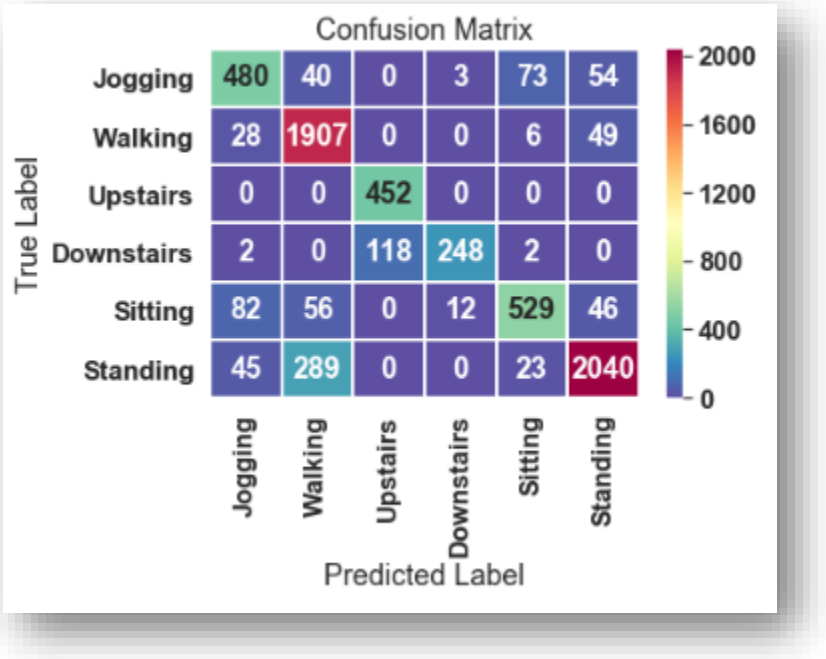

Figure G: Confusion Matrix for Each Action Session (Pure CNN)

❖ LSTM

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 the relative context, hence their name. Since what a user did before/after going up or downstairs might be relevant to which action they were performing, LSTM's were used over RNN's to include such a 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.

Two LSTM layers were added right before the dropout layer, once in addition to the 3^{rd}/4^{th} CNN layer and then replacing them entirely. The theory behind this is allowing the model, after most generalizations have been made, to maintain a memory of the overarching themes behind the different user actions. Figure H1 also shows that keeping all original CNN layers while adding the two LSTM layers found a better accuracy/loss at an even earlier epoch.

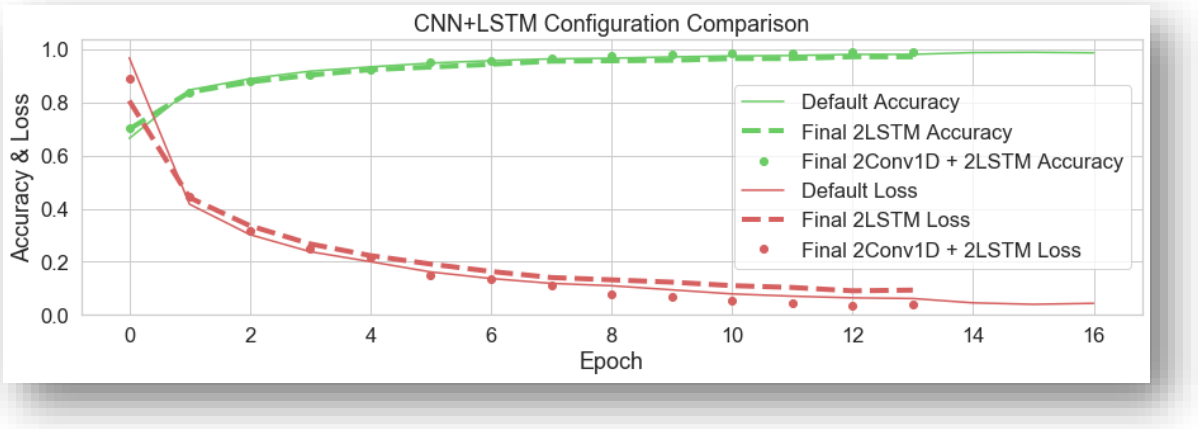

Figure H1: Effect of CNN+LSTM Configuration on Model Performance

These improvements in accuracy are more-or-less marginal considering the early Conv1D layers had already done most of the work by the time the LSTM's further refine the convolved features. That is not to say, though, that a CNN + LSTM architecture wouldn't prove beneficial to deciphering more nuanced actions for classification with the addition of more sensors. This was the case for a 2016 study which found substantial accuracy with this model and which the idea for CNN+LSTM came from, though based on a more complex data set called OPPORTUNITY (Ordóñez & Roggen, 2016).

A more 1:1 comparison of the different approaches of LSTM's vs. Conv1D's was done by using the same CNN architecture except replacing all Conv1D layers with LSTM's. The remaining question is if LSTM layers alone can bring the unique signatures of each action into context for the model's dense layer to show little to no confusion. This full layer representation is shown in Figure H2.

Figure H2: CNN Output & Activation by Layer & Action

Once again, the downstairs action's activation is investigated after the first layer in Figure H3. Since the features are not convolved at this point, the kernel of the x-axis shows us what feature explicitly caused higher or lower activity. Interestingly, the LSTM layer doesn't see a large spike in activity when the user suddenly decelerates (time period ~67). Since this is all later layers can interpret, there will likely be a high degree of confusion with the upstairs action (just as we saw for the CNN).

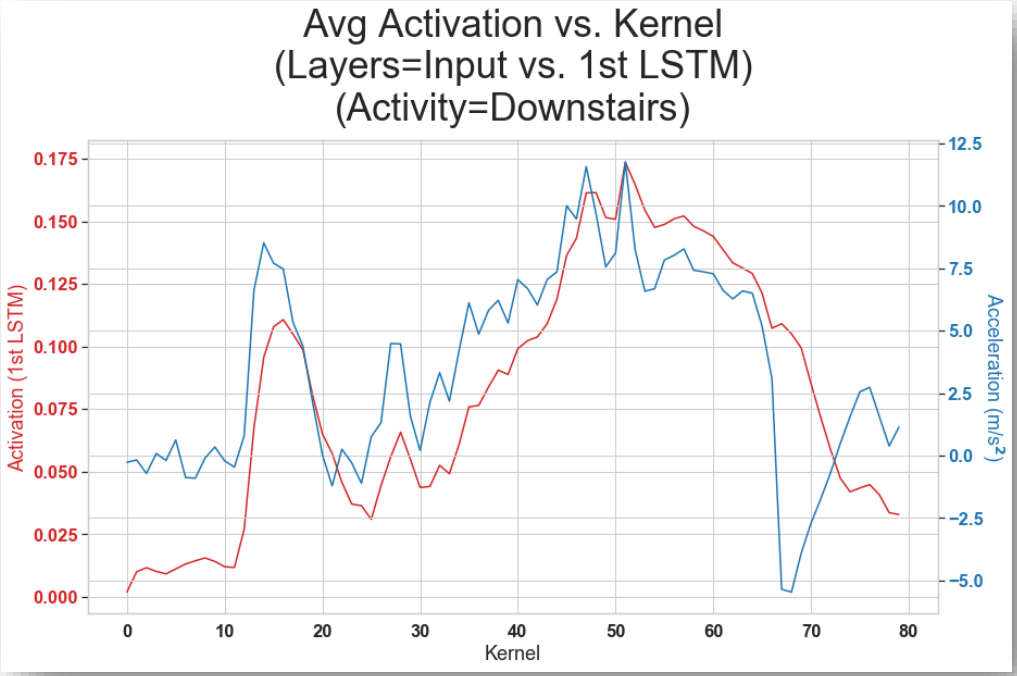

Figure H3: LSTM Model Downstairs Input vs. First Layer Activation

Figure H4 shows the final LSTM activation for the downstairs action through this model. It is very similar to the CNN's final interpretation of the upstairs action. Since average pooling with a kernel width of 2 was used, the kernels available have been reduced by half. A small

shoulder, possibly acceleration on the first step down, also makes an appearance. Figure H5 shows how the final LSTM layer represents the upstairs action.

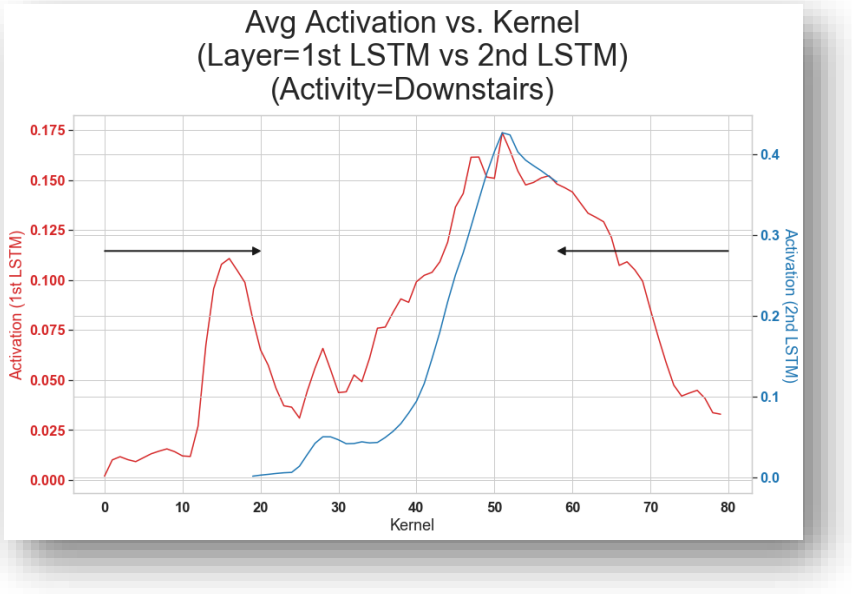

Figure H4: LSTM Model Downstairs First LSTM vs. Final LSTM Activation

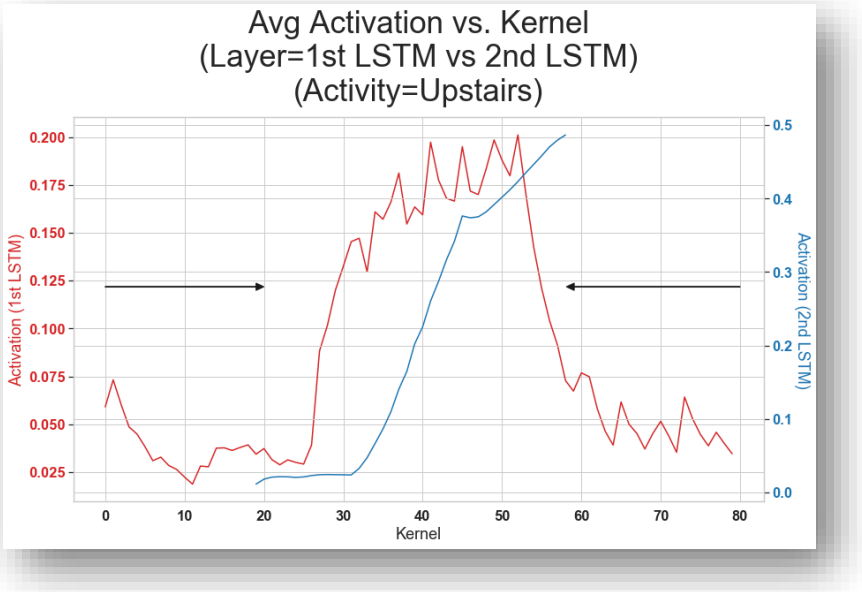

Figure H5: LSTM Model Upstairs First LSTM vs. Final LSTM Activation

LSTM's show a great deal of smoothing out the noise even by the first layer. The similarity of the second LSTM output between upstairs/downstairs might imply the 'memory' in LSTM was still not great enough to keep in context what the user had done at the beginning of the stairs. In some ways, it could be theoretically possible that LSTM's could outperform CNN's for very short actions such that LSTM's could bring them into context whereas CNN's would, at fixed sampling Hz, lack enough kernels to separate the features from the noise. It also reiterates why CNN + LSTM was a good fit for the cited OPPORTUNITY study: Conv1D for accelerometer data is good about pointing out the important disparate features and LSTM is good at bringing them quickly into context.

❖ LSTM Performance

A confusion matrix for this LSTM model is shown in Figure I. The overall accuracy for the LSTM model was 75% on the test data. The LSTM model incorrectly classified only 16% of the upstairs actions as downstairs whereas the CNN model incorrectly classified 32% which was quite a large improvement. Jogging, typically an action of longer duration, was almost always mistaken as sitting or standing. Length for LSTM context alone could not be the sole factor of underperformance, however, since sitting was confused for standing often. One possibility is that sitting and standing features, to be significant, might rely on the temporal context of the time both before and after the significant features is complete. Currently, the LSTM model has only context temporally in one direction. As we've seen with stairs actions, the important feature is often a sudden acceleration/deceleration. Next, we will discuss important improvements to this simplistic LSTM architecture and how that time context can be improved.

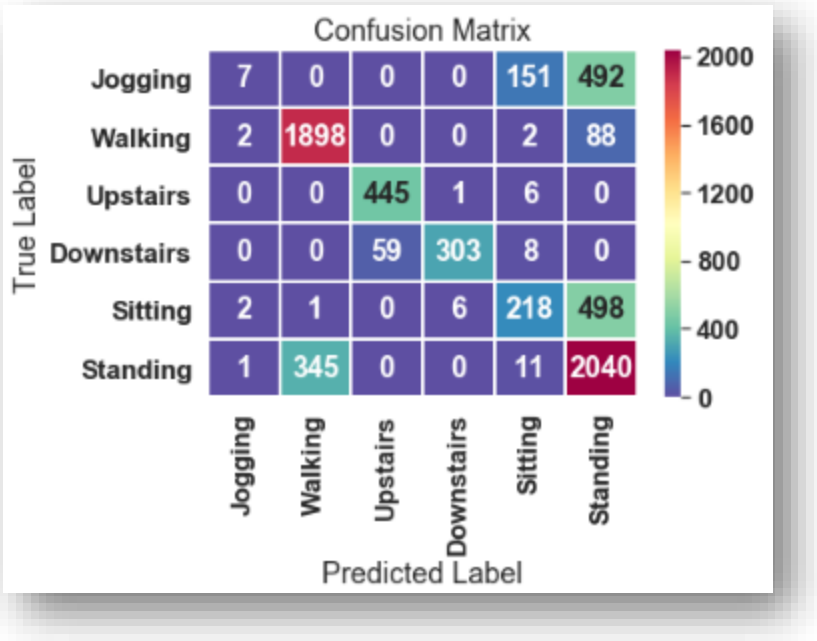

Figure I: LSTM Model Upstairs First LSTM vs. Final LSTM Activation

❖ The Vanishing Gradient Problem

The 1:1 comparison of LSTM's against CNN's showed for this data set that LSTM's are susceptible to the vanishing gradient problem that RNN's and deep learning, in general, can be susceptible to. This may not have been a truly fair comparison since LSTM's were put into an architecture found optimal for CNN's. Citing the apparent weakness of LSTM's due to lack of temporal context, several literature sources of alternative methods were found. An example is shown in Figure J of an alternate architecture that utilizes only LSTM's to achieve high accuracy with human activity recognition (Zhao, Yang, Chevalier, Xu, & Zhang, 2018).

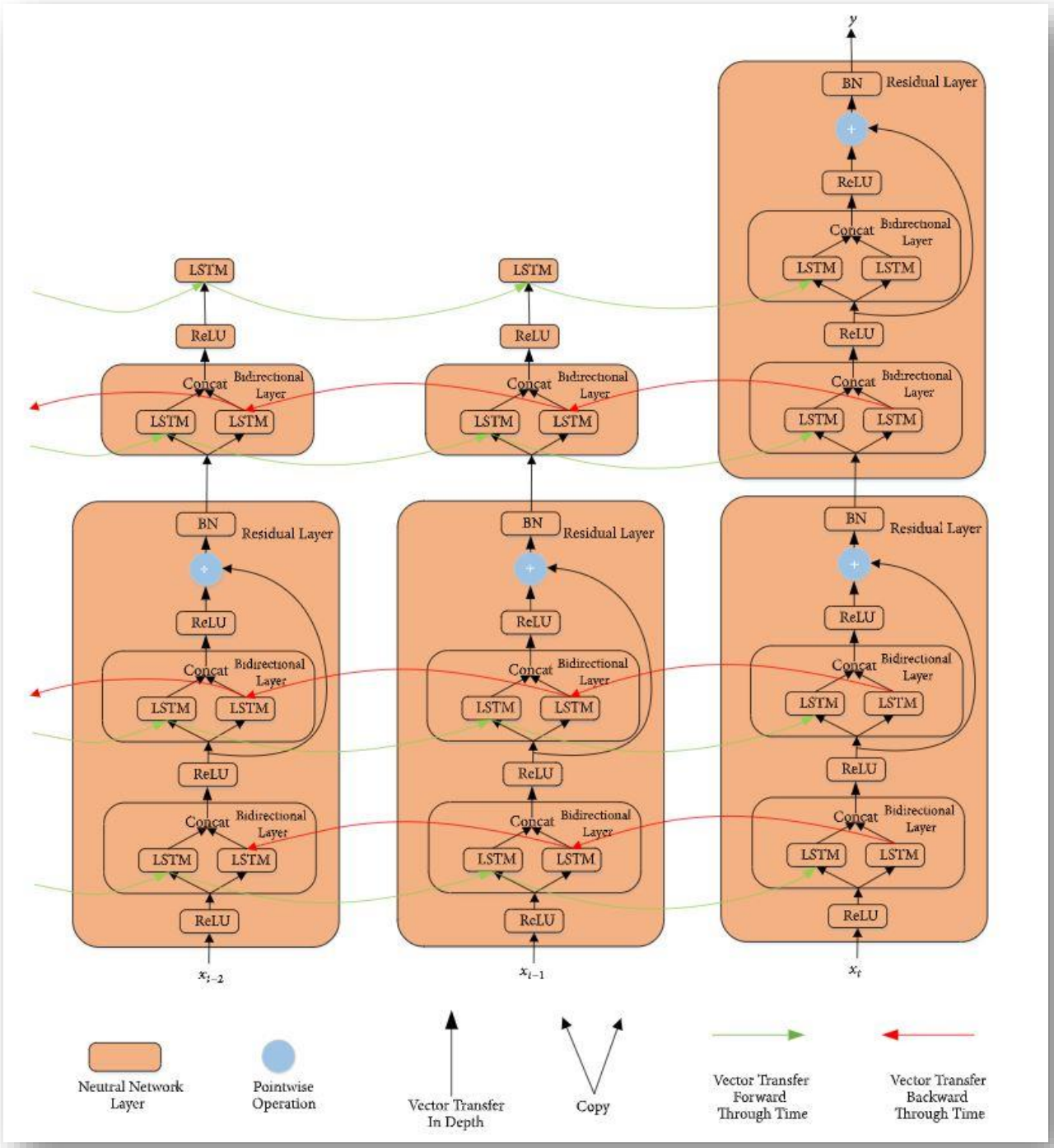

Figure J: Res-Bidir-LSTM Network Architecture

This architecture employs two concepts that have not yet been discussed: residual & bidirectional. The bidirectional set-up in Figure J, as illustrated by the green & red time arrows,

shows how a model can resolve issues of understanding feature context both forward and backward in time. The residual layer allows LSTM activation to shortcut the linear operator as well as ReLU non-linearity to effectively feed-forward its understanding of the feature. This allows the training of much deeper neural networks without suffering the typical penalty of vanishing gradient. Memory for any individual LSTM now doesn't need to have to do nearly as much work by both having the feed-backward and feed-forward time context. Additionally, the model suffers less from overly complex features by intermittently passing them beyond ReLU. A feature that was previously unlearnable, over epochs of adjustment, may now in context become learnable with this modification.

❖ Further Research

Quite a few findings in this paper showed the CNN was relatively adept at this dataset but not necessarily the best architecture for all motion-sensing phone applications. Factors this model was dependent on include:

- The quality of the action performed
- How long the action was performed
- The lack of additional context for those actions (e.g. GPS/time-of-day)
- The sampling Hz
- The chosen actions themselves
- The WISDM study only utilized the three-axis system for this data set

Besides factors outside the control of this analysis, several explorations could still be done to explore the fit of better models with the data & tools available. Items explored which could be explored further include:

- Different pooling configurations
- Different LSTM + CNN (e.g. alternative layer types) configurations
- Different kernel/filter configurations

Items not explored in-depth for this study include:

- Different activation functions
- Iterating different learning rates (static or dynamic over epoch)
- Utilizing different time intervals
- Setting different criteria for model convergence
- Different layer types besides convolutional/recurrent
- Further analysis on whether false positives/negatives were isolated to specific users (clustering)

All these factors will clearly not pose equal weight on the model, either. Considering the amount of data and the high degree of accuracy achieved on the WISDM data set by this CNN model and others, it is safe to say the data quality including sampling techniques are not to model detriment. Based on non-uniformity in the activations of the nodes in each heatmap, the learning rate, time interval, and activation functions are likely good candidates. This was also shown later in the average activation over layers showing significant differences in learned features. Likely the most improvements in further research would be seen in architectural

modifications. As many of the literature sources cited in this article show, certain configurations require less parameter tuning and achieve higher accuracy with less need for optimization.

This study and those like it confirm that deep learning is both a viable and dynamic way to achieve high accuracy on determining user behavior from accelerometer data. If the features and their actions are simple enough, CNN's are good candidates. For more complex features in the same time sampling limitations, neural net architectures have dynamic solutions by:

- Using LSTM's, time limitations previously encountered by CNN's can be overcome for even similar actions (e.g. stairs)
- Using LSTM's with residuals, Zhao et. al have also found a way to utilize this type of layer without suffering from the vanishing gradient problem
- Convolving the features into context for subsequent LSTM's to fully further learn the features as demonstrated by Ordóñez & Roggen

Even unexpected behavior in the exploration of these architectures offered up new possibilities. By learning why & how LSTM's were weak in certain situations, different architectural designs have been created to addresses these shortfalls. Each solution is not just an improvement on LSTM's; rather, each improvement has led to specialized architectures that each excel in very specific categories. By continuing to discover the best architecture for different data sets, a precedent is set for which of these use-cases correlate to each accelerometer set-up and actions performed. From there, the goal is to apply this deeper understanding of architectural pertinence to other data sets and use-cases, even those that are entirely dissimilar from this study.

References

Ackermann, N. (2018, September 4). Introduction to 1D Convolutional Neural Networks in Keras for Time Sequences. Retrieved October 25, 2019, from https://mc.ai/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences/.

Bengio, Y., Simard, P., & Frasconi, P. (1994, March). Learning long-term dependencies with gradient descent is difficult. Retrieved October 25, 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. Retrieved October 25, 2019, from http://www.cis.fordham.edu/wisdm/includes/files/sensorKDD-2010.pdf.

Ordóñez, F. J., & Roggen, D. (2016, January 18). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. Retrieved October 25, 2019, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4732148/.

Zhao, Y., Yang, R., Chevalier, G., Xu, X., & Zhang, Z. (2018, December 30). Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors. Retrieved October 25, 2019, from https://www.hindawi.com/journals/mpe/2018/7316954/.