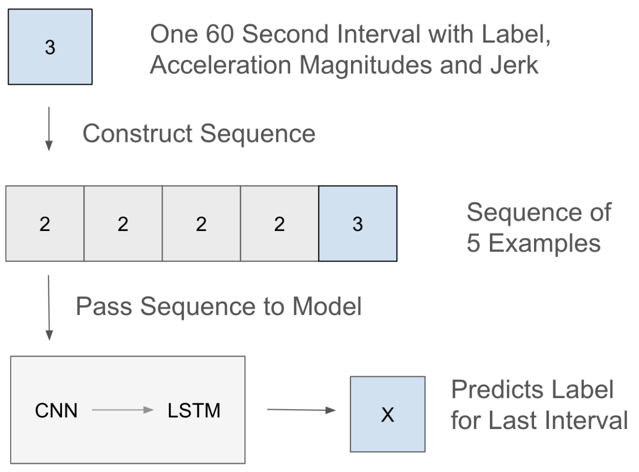
Study 2: HAR from Phone Inertial Sensor Data

Algorithm

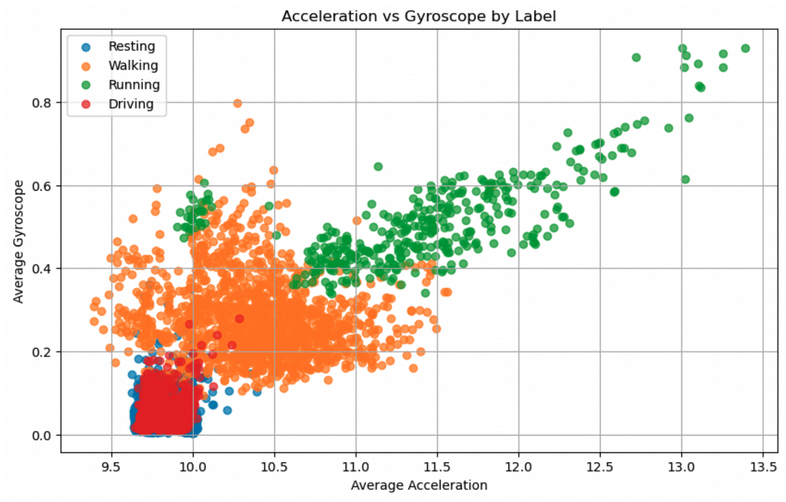
We took a deep learning approach where the model uses a CNN to extract features and passes the extracted features into a LSTM for temporal dependency learning. The first preprocessing step is to find the L2 norm of the acceleration and gyroscopic, and the jerk. For temporal dependencies to work, the algorithm then constructs sequences of length 5 that were composed of 5 consecutive training examples using the label in the example to label the sequence. Finally, each sequence is passed to the model for feature extraction and prediction using 60 extracted features.

The CNN uses 3 convolutional layers with ReLU activation and max pooling with kernel size 2, stride 2, and dilation 1. No padding for the pooling is needed as the strides match the channels nicely. The first convolution layer takes 3 input channels (one for each acceleration direction) and 16 output channels with a kernel size of 3, stride 1, and padding 1. The second convolutional layer outputs 32 channels, and the third layer outputs 64 channels, all with the same hyperparameters as the first layer. Dropout was used after each pooling with probability 0.5. The output is flattened and passed to the LSTM. The LSTM model has 2 layers that take in 448 to 128, and a dense 128 channel to the 4 possible labels as the channel sizes. The figure shows the progression in the algorithm.


3
One 60 Second Interval with Label, Acceleration Magnitudes and Jerk
Construct Sequence
2
2
2
2
3
Sequence of 5 Examples
Pass Sequence to Model
CNN
LSTM
X
Predicts Label for Last Interval

Pre-Processing

We explored the data by graphing the average L2 norm of each type of acceleration as shown:



As expected, the driving and resting labels are shown to be similar since the average acceleration should be very small. From this graph, we can tell that classifying resting and driving would be the most difficult task.

We believed that solely using the magnitude for each sensor type would be sufficient. This is because for any type of travel, the person is expected to go “forwards,” (people don’t usually travel like crabs) and so the direction of acceleration is not necessarily the key feature compared to the patterns of acceleration magnitude. For gyroscopic acceleration, the graph shows that running typically has an average gyroscopic acceleration greater than or around 0.4 units, and so we decided that the model would be able to distinguish between running and walking just from the magnitude patterns over time. As a result, we simplified the 6 given files into just 2 datasets: the magnitude of both the acceleration and the gyroscopic acceleration. Using a hint in class, we also calculated the jerk magnitude, the change in acceleration over time, and used that as our third input channel as it can help classify driving from resting.

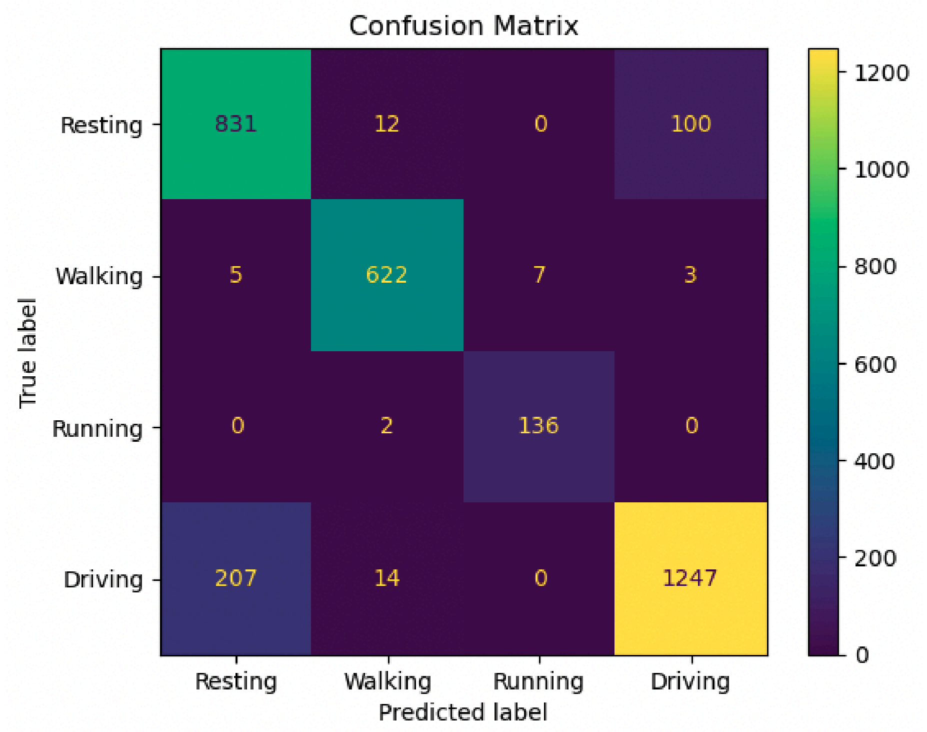
To obtain additional data, we synthesized our data by first reorganizing the given data into groups by label and retaining their original order. To generate new data, we split 2 consecutive training examples in half and join the split halves together to form a single new training example with the same label (we only have one new example because the order matters). This splitting and joining is done for each acceleration direction for a total of 6 new files of generated data. The data is then split into training and testing data and added to the end of the training data and testing data made from the original dataset. We thought about adding noise, but decided against it as the synthesized data likely carried the original noise from the combined examples over. The purpose of this data synthesis process was to have more data for a held-out test set, but also account for the lower number of examples with the running label, even if the proportion of running examples compared to the other examples is unchanged.

In order for the temporal relationships to be learned in the LSTM, we create sequences of examples similar to the sliding window segmentation in Hammerla 2015 (pg 1044). The training sequences are created by padding the first few examples with zeros, and then adding the features of the examples to a sequence. The sequence length parameter is tuned to 5 and the label of the sequence is the last in example in the sequence. The sequence of 5 is similar to 300 second intervals but with a single label. Each example has a sequence created for it and the first part of the beginning sequences may be 0’s due to the padding. The label for each sequence is the label for the last example due to the padding choice. For testing data, the same thing is done but without labels. The purpose of this step is to help classify the resting and driving label. By considering sequences, or rather the time intervals from previous intervals as a whole, the neural network is able to extract features relating the current label and previous label, which then allows the LSTM to use those patterns to better classify driving and resting.

Testing

After the exploratory data analysis, we tested the data with a CNN model. Taking inspiration from the literature, we looked at the methods Hammerla used to classify the PAMAP2 dataset in the 2016 paper as that dataset appeared to be most similar to our dataset (pg 1534, 1538). The paper graphics show that their CNN uses max pooling and dense layers to classify activity. Our model used 3 convolution layers each with ReLU activation, max pooling, and dropout layers. The first convolution layer takes in two channels and outputs 16 channels. The 2 in-channels are for each acceleration magnitude, and the output is 16. The padding is 1 and the stride length is 1 with kernel size 3. The same is done with the second layer that outputs 32 channels, and the third layer outputting 64 layers. In between each layer, there is ReLu activation, max pooling with kernel size 2 and stride length 2, and dropout with probability of being dropped at 0.3. The purpose of these layers is to regularize the network, preventing overfitting. This result is then flattened and fed into a dense layer with ReLU activation and dropout at 0.5, and then a dense output layer. The first dense layer takes in 448 channels (from the flattening) and outputs 128 and the output layer takes the 128 and outputs the 4 possible labels.

This neural network resulted in this confusion matrix after applying it on a held out test set on the last 30% of the original data and the synthesized data:



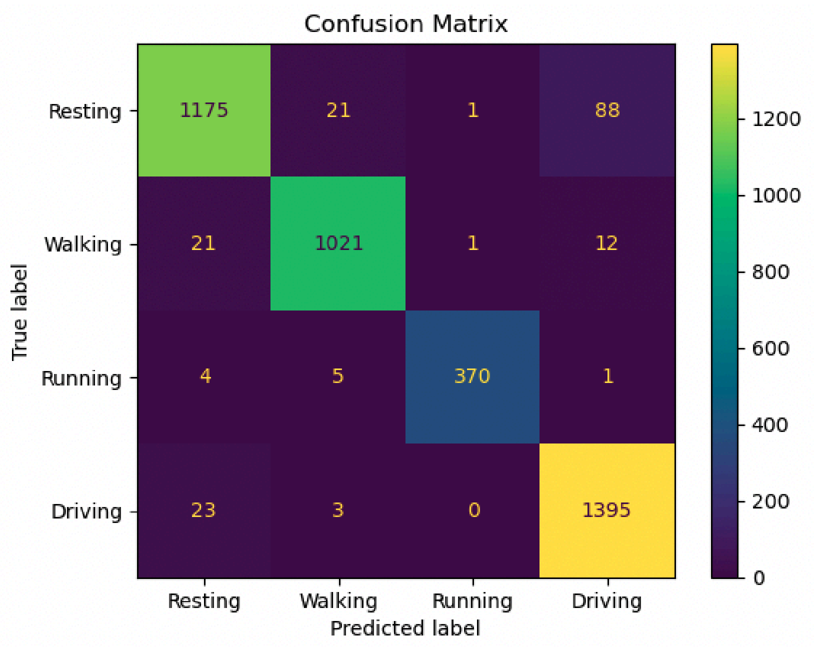
Evidently, the network misclassified driving and resting as we expected from exploratory data analysis results. To account for this, we take the suggestion in class and pass the convolution result to a LSTM. There are two reasons for choosing an LSTM. Firstly, the 2016 Hammerla shows that the LSTM accuracy for the PAMAP2 dataset is .929—a 0.008 difference from the

CNN, showing that the LSTM model can sufficiently classify HAR data. Secondly, the use of LSTM allows for the consideration of previous examples which is suitable for this dataset as the data follows a sequential ordering. For this to happen, we crafted sequences of length 5 for the examples as discussed in the prior section.

Our first attempt at this combined approach uses a similar model architecture as our final algorithm, except that we had 2 convolution layers and had 2 input channels for the first convolutional layer as it represents the magnitude of the linear acceleration and gyroscopic acceleration. Since there were less layers, the output channels were also halved, leading us to removing the dropout layers between. Training was done using a GPU with 60 epochs (early stoppage) and batch size of 32. The sequences were shuffled as the sequences themselves retain the previous ordering up to the sequence length. Standard cross-entropy loss and the adam optimizer was used and we achieved $F1_{micro} = 0.915$ and $F1_{macro} = 0.936$.

Seeing improvements compared to the first dense CNN, we decided to further this model. We attempted to re-add in the directional data. The reasoning is because the CNN acts as the feature extraction, and we are limiting the extraction by focusing on solely the magnitudes. In other words, we are doing a poorer job at feature extraction than the CNN model by only giving it the L2 norm. In this case, we might as well pass all the features into the CNN model to extract the most data. Another reason is because the neural network is not as large as the first dense model. Testing this new model using solely the "train_2" data files as the test set, our results were as follows: $F1_{micro} = 0.978$, $F1_{macro} = 0.980$.

Yet, we still believe that the L2 norms were sufficient features to capture the acceleration patterns. At the suggestion of considering integrating acceleration to get velocity in class, we implemented jerk, the change in acceleration over time. By including jerk as a new input channel, we add a convolution layer and dropout layers at 0.5, believing this would help with feature extraction of the CNN. We tuned that 60 extracted features from the sequences worked well and conducted training on 50 epochs. The results for this new model using a held-out test set is as follows: $F1_{micro} = 0.956$, $F1_{macro} = 0.962$.



While having slightly lower F1 scores compared to using the 6 input channel model, we believe this model has a lower risk of overfitting as it blends the potential noise in the directional acceleration into the L2 norm. As a result we decided to choose this model instead.

Recommendation

A limitation to the algorithm is that the training is done only on one person. Assuming this person is carrying the phone on their body, the acceleration patterns are expected to differ from someone who carries the phone in a purse or backpack, or even in a different pocket in their pants, not to mention the variance due to walking gaits. In addition, the sequence length parameter need to be tuned depending on the occupation of the subject as a result of the frequency of their label changes (ie. garbage collector versus pilot). For this reason, it is best to train the algorithm on a single person, and deploy it on the same person as well.

The algorithm uses sequences so the sequences must be constructed and the assumption is that the examples composed of the sequences are mostly consecutive. In a real-time setting, the algorithm does not need to access future data, but the most recent past data to construct the sequences and so no change is necessary for it to function as long as the data is entered consecutively and is reasonably within the same timeframe. However, the algorithm would not be expected to perform well if the sequence construction is made of examples that are too far apart. For instance, if the algorithm tries to predict the label at 8:03 am using the examples from 10:00 pm to 10:04 pm, it may not correctly classify the label. One way to deal with this is to have the data include a timestamp, and if the times are more than a certain threshold apart, the sequence would be padded with 0's. If limited to the current data, we could potentially pad a sequence length *l* of 0's between the different labels, increasing assurance that the sequences are not dependent on irrelevant time intervals.

The running label appears less than the other labels, which suggests that $F1_{macro}$ may be a better metric than $F1_{micro}$ as it considers the imbalance in class data (Hammerla 2015). However, since the model had less trouble misclassifying the running label compared to classifying driving and resting, ROC or Macro Precision-Recall curves is an alternative accuracy measure to visualize the correctness of the model.

References

Hammerla, N. Y., Halloran, S., & Plötz, T. (2016). Deep, convolutional, and recurrent models for human activity recognition using wearables. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (pp. 1533-1540).

Hammerla, N. Y., & Plötz, T. (2015). Let's (not) stick together: Pairwise similarity biases cross-validation in activity recognition. In Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (pp. 1041-1051).