A Deep Learning Approach for Human Activity Recognition
Project Category: Other (Time-Series Classification)

Susana Benavidez
susana@cs.stanford.edu

Derek McCreight
dmccreig@stanford.edu

Abstract

The hardware and sensors in smartphones and wearable devices are becoming more powerful and precise every year; the valuable data these sensors collect can be used for more precise Human Activity Recognition. In this paper, we explore implementing both Convolutional Neural Networks and LSTMs to classify 18 unique activities ranging from eating chips to dribbling a ball from labelled, time-series gyroscope and accelerometer data collected from a smartwatch and smartphone. All of our work is located in our public repo here, and the processed data can be accessed here.

1 Introduction

Wearable devices and smartphones are ubiquitous, and a majority of these devices contain many sensors, including Inertial Measurement Units **(IMUs)** such as gyroscopes and accelerometers. As these sensors have become cheaper and more available, an increasing number of people have 'always-on' accelerometers and IMU sensors active. There are an increasing number of machine learning applications for using this data, given how much information can be inferred from them. For example, IMU data can be used for ML-based gesture recognition, like with the Raise to Speak feature on the Apple Watch.

This data could also be used to perform human activity recognition **(HAR)**, to recognize the motion characteristics of the user. This has several applications, for example, in healthcare, to measure the activity and fitness characteristics of the user. Some devices do this for fitness tracking, but the potential for activity recognition extends beyond just exercise; the utility of HAR becomes more apparent when it can be used to classify many more everyday activities, such as logging when you brush your teeth, or automatically estimating your food intake based on estimation of when you are eating and drinking.

Using IMU data for this purpose is also often preferable to other modalities; it is privacy-preserving since there is less user-identifiable data compared to audio or camera data, and IMUs tend to be less computationally and power-intensive than other sensors such as cameras and microphones.

2 Related work

Another common method that is proposed to extract frequency-domain features using MFCCs or Spectrograms on the accelerometer data. However, these feature-based approaches require an intricate and extensive knowledge of the domain, and often are time-consuming to both identify and compute.(1) They are also brittle, and it is difficult to optimize the features end-to-end, since they are the first input to the model (4). As the number of motion activities scale, so does the effort in identifying relevant features. Some previous work (2) has investigated using deep learning for HAR, typically with one-dimensional convolutional neural networks **(CNNs)** (7).

The experiments by Bevilacqua (3) identify a deep learning approach based on CNNs, but are based on exercises rather than common activities. Their dataset was derived from specially placed sensors on the lower half of the body, which makes it less applicable to a large user base, as most people just have smartphones or smart watches.

Weiss (5) opted to use five distinct algorithms: Random Forests, J48 decision trees, B3 instance-based learning, Naive Bayes, and multi-layer perceptron. Using these algorithms, Weiss was able to obtain an overall accuracy rate of 25.3% with the phone accelerometer data, and 64% accuracy with the watch accelerometer data.

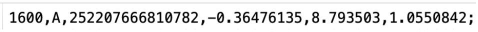
3 Dataset and Features

We are using a brand new **WISDM** dataset (5), released this year by Fordham University which can be downloaded online here. The dataset was created from 15 different participants, each of whom wore a smartwatch on their left wrist, and either a Google Nexus or Samsung Galaxy S5 in their pocket. The phone was placed in their right pocket upright, with the screen of the phone facing away from the body. Each participant performed 18 different activities with each device separately for 4 consecutive minutes each. The full list of activities were: walking, jogging, climbing stairs, sitting, standing, typing, brushing teeth, eating soup, eating chips, eating pasta, drinking from a cup, eating a sandwich, kicking a soccer ball, playing catch with a tennis ball, dribbling a basketball, writing, clapping, and folding clothes.

What makes this dataset fairly unique is that the activities go beyond just motion activities; for example, the dataset contains categories like "brushing teeth" and "eating chips". This gives it applications beyond most other datasets which focus on exercise or motion only.

The gyroscope and accelerometer tri-axial data was sampled at a rate of 20Hz, where each sample contains scalar data about the devices current position in space. Each sample is of the form:

[**Participant ID** (beginning at 1600)], [**timestamp** (unix-based)], [**x value**], [**y value**], [**z value**] as shown below in Figure 2.



```
1600,A,252207666810782,-0.36476135,8.793503,1.0550842;
```

Figure 1: Example of accelerometer/gyroscope data

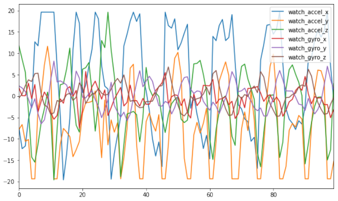


Figure 2: Phone Accel/Gyro Data when Jogging

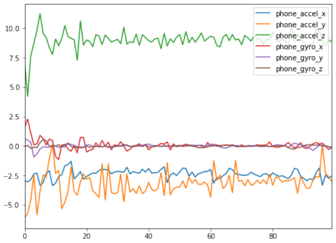


Figure 3: Phone Accel/Gyro Data when eating sandwich

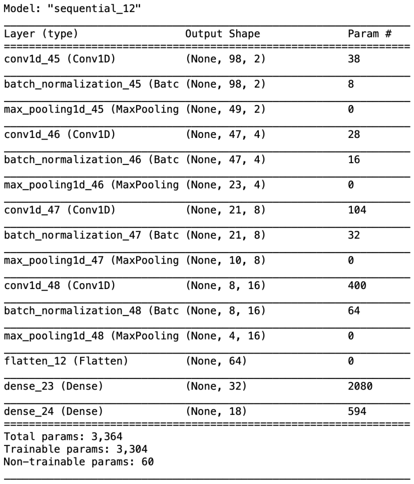
Since the dataset has a hierarchical file structure, (each participant has 18 folders for each activity) we first aggregated the data for each participant together into 15 different data-frames. Next, we merged all phone and watch data from each participant together. We then created sliding windows of approximately 2 seconds with an equivalent stride to slice the sensor data into different parts. Finally, we combined the separated windows for both the phone and watch data.

4 Methodology

More traditional algorithms have been applied to activity recognition, however as previously mentioned this requires an extensive amount of modelling and creating higher level features to represent the problem, and oftentimes this may not be an accurate representation of the problem leading to inaccuracies at prediction time. CNN/LSTM have the capacity to learn temporal dependencies between outputs from the phone/watch sensors, therefore we have opted to implement both an LSTM and CNN architecture for our task. We are using Weiss' (5) best performing algorithm as our baseline since he also trained his models on the WISDM dataset.

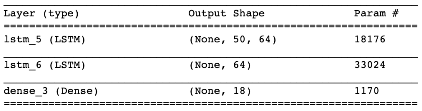
5 Model Architecture

Our Convolutional Neural Network consists of 4 Conv1D layers, accompanied with Batch Normalization and Max Pooling layers and lastly a flatten and two dense layers as shown below:



```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_45 (Conv1D)           (None, 98, 2)             38        
_________________________________________________________________
batch_normalization_45 (Batc (None, 98, 2)             8         
_________________________________________________________________
max_pooling1d_45 (MaxPooling (None, 49, 2)             0         
_________________________________________________________________
conv1d_46 (Conv1D)           (None, 47, 4)             28        
_________________________________________________________________
batch_normalization_46 (Batc (None, 47, 4)             16        
_________________________________________________________________
max_pooling1d_46 (MaxPooling (None, 23, 4)             0         
_________________________________________________________________
conv1d_47 (Conv1D)           (None, 21, 8)             104       
_________________________________________________________________
batch_normalization_47 (Batc (None, 21, 8)             32        
_________________________________________________________________
max_pooling1d_47 (MaxPooling (None, 10, 8)             0         
_________________________________________________________________
conv1d_48 (Conv1D)           (None, 8, 16)             400       
_________________________________________________________________
batch_normalization_48 (Batc (None, 8, 16)             64        
_________________________________________________________________
max_pooling1d_48 (MaxPooling (None, 4, 16)             0         
_________________________________________________________________
flatten_12 (Flatten)         (None, 64)                0         
_________________________________________________________________
dense_23 (Dense)             (None, 32)                2080      
_________________________________________________________________
dense_24 (Dense)             (None, 18)                594       
=================================================================
Total params: 3,364
Trainable params: 3,304
Non-trainable params: 60
_________________________________________________________________
```

Figure 4: 4ConvLayer Neural Network Architecture



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (None, 50, 64)            18176     
_________________________________________________________________
lstm_6 (LSTM)                (None, 64)                33024     
_________________________________________________________________
dense_3 (Dense)              (None, 18)                1170      
=================================================================
```

Figure 5: Stacked-LSTM Model Structure

For the LSTM structure, we decided to use a stacked-LSTM structure. We chose to stack multiple recurrent states with multiple memory cells because it allows the model to determine more complex abstractions from the input data. Specifically, we have two LSTM layers, followed by a Dense Layer.

Since we are performing classification on our data, the **Cross Entropy Loss** is a natural loss function to use. For classification problems it is equivalent to the Maximum Likelihood Estimation (MLE). The multi-class Cross Entropy Loss is defined as follows:

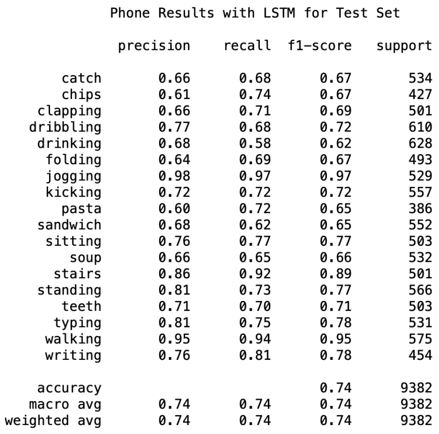
$$J = \frac{1}{N}(\sum_{i=1}^{N} y_i - log(\hat{y_i}))$$

where $\hat{y_i}$ is the predicted label for the ith training example, y_i is the true desired label, and N is the number of training examples.

6 Experiments/Results/Discussion

Comparing LSTM and CNN on both watch and phone data sets, we see that the LSTM performs better than the CNN with 79% vs. 72% accuracy and 74% vs. 50% for each respective dataset. The resulting confusion matrices show that the models on the phone dataset struggle to differentiate between activities that require similar hand movements such as eating chips versus eating soup. Similarly, the models had a difficult time distinguishing between catching, kicking, and dribbling on the watch dataset likely because the hand movements aren't what most distinguishes these activities.

It's worth noting that despite these difficulties, the model still manages to greatly improve upon our baseline from Weiss. Additionally, if we were to group all eating activities together, the accuracy of both models would be much higher.



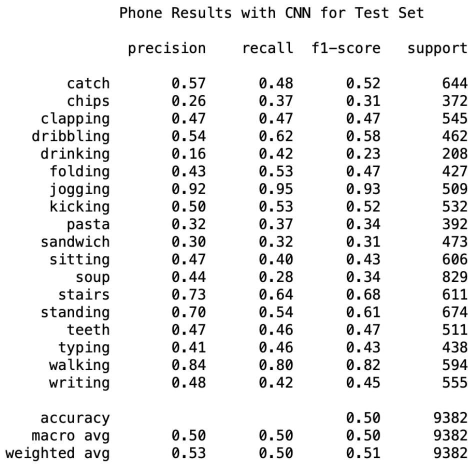
```
              Phone Results with LSTM for Test Set

               precision    recall  f1-score   support

       catch       0.66      0.68      0.67       534
       chips       0.61      0.74      0.67       427
    clapping       0.66      0.71      0.69       501
   dribbling       0.77      0.68      0.72       610
    drinking       0.68      0.58      0.62       628
     folding       0.64      0.69      0.67       493
     jogging       0.98      0.97      0.97       529
     kicking       0.72      0.72      0.72       557
       pasta       0.60      0.72      0.65       386
    sandwich       0.68      0.62      0.65       552
     sitting       0.76      0.77      0.77       503
        soup       0.66      0.65      0.66       532
      stairs       0.86      0.92      0.89       501
    standing       0.81      0.73      0.77       566
       teeth       0.71      0.70      0.71       503
      typing       0.81      0.75      0.78       531
     walking       0.95      0.94      0.95       575
     writing       0.76      0.81      0.78       454

    accuracy                           0.74      9382
   macro avg       0.74      0.74      0.74      9382
weighted avg       0.74      0.74      0.74      9382
```

Figure 6: LSTM results for Phone



```
              Phone Results with CNN for Test Set

               precision    recall  f1-score   support

       catch       0.57      0.48      0.52       644
       chips       0.26      0.37      0.31       372
    clapping       0.47      0.47      0.47       545
   dribbling       0.54      0.62      0.58       462
    drinking       0.16      0.42      0.23       208
     folding       0.43      0.53      0.47       427
     jogging       0.92      0.95      0.93       509
     kicking       0.50      0.53      0.52       532
       pasta       0.32      0.37      0.34       392
    sandwich       0.30      0.32      0.31       473
     sitting       0.47      0.40      0.43       606
        soup       0.44      0.28      0.34       829
      stairs       0.73      0.64      0.68       611
    standing       0.70      0.54      0.61       674
       teeth       0.47      0.46      0.47       511
      typing       0.41      0.46      0.43       438
     walking       0.84      0.80      0.82       594
     writing       0.48      0.42      0.45       555

    accuracy                           0.50      9382
   macro avg       0.50      0.50      0.50      9382
weighted avg       0.53      0.50      0.51      9382
```

Figure 7: CNN results for Phone

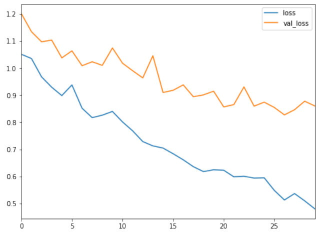


Figure 8: LSTM loss for Phone

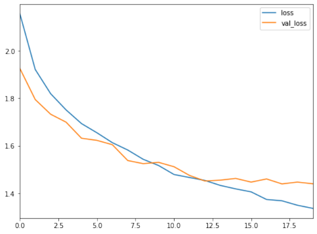


Figure 9: CNN loss for Phone

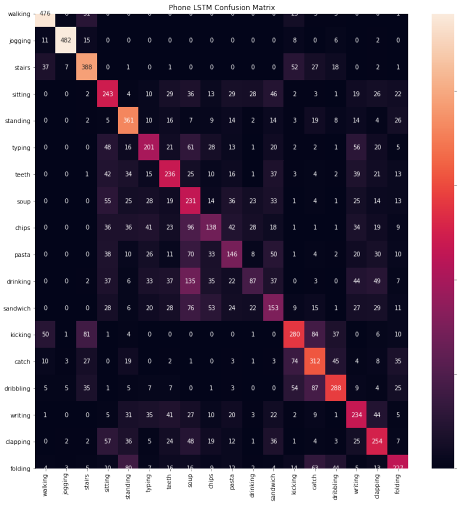


Figure 10: LSTM Confusion Matrix for Phone

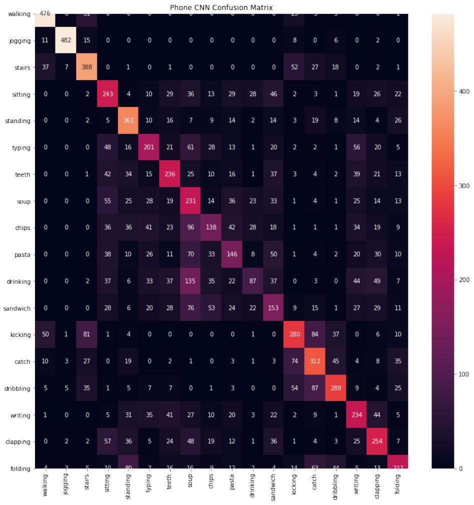
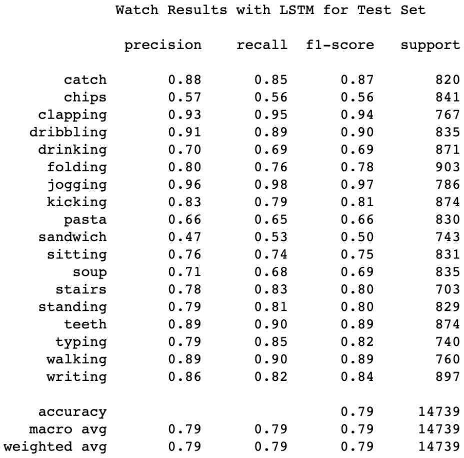


Figure 11: CNN Confusion Matrix for Phone



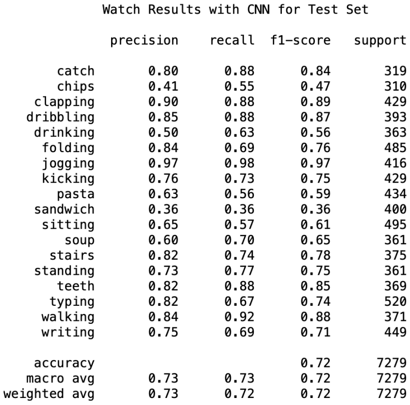
```
              Watch Results with LSTM for Test Set

               precision    recall  f1-score   support

       catch       0.88      0.85      0.87       820
       chips       0.57      0.56      0.56       841
    clapping       0.93      0.95      0.94       767
   dribbling       0.91      0.89      0.90       835
    drinking       0.70      0.69      0.69       871
     folding       0.80      0.76      0.78       903
     jogging       0.96      0.98      0.97       786
     kicking       0.83      0.79      0.81       874
       pasta       0.66      0.65      0.66       830
    sandwich       0.47      0.53      0.50       743
     sitting       0.76      0.74      0.75       831
        soup       0.71      0.68      0.69       835
      stairs       0.78      0.83      0.80       703
    standing       0.79      0.81      0.80       829
       teeth       0.89      0.90      0.89       874
      typing       0.79      0.85      0.82       740
     walking       0.89      0.90      0.89       760
     writing       0.86      0.82      0.84       897

    accuracy                           0.79     14739
   macro avg       0.79      0.79      0.79     14739
weighted avg       0.79      0.79      0.79     14739
```

Figure 12: LSTM results for Watch



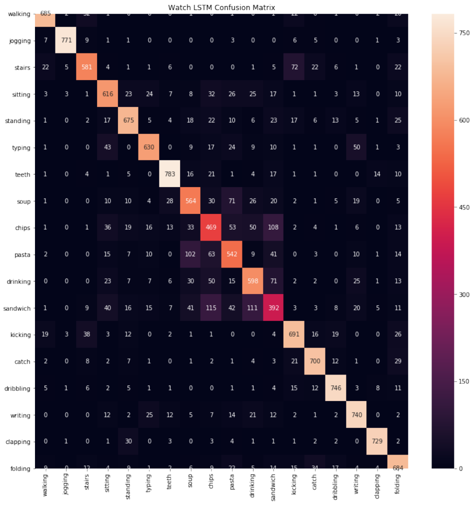
```
              Watch Results with CNN for Test Set

               precision    recall  f1-score   support

       catch       0.80      0.88      0.84       319
       chips       0.41      0.55      0.47       310
    clapping       0.90      0.88      0.89       429
   dribbling       0.85      0.88      0.87       393
    drinking       0.50      0.63      0.56       363
     folding       0.84      0.69      0.76       485
     jogging       0.97      0.98      0.97       416
     kicking       0.76      0.73      0.75       429
       pasta       0.63      0.56      0.59       434
    sandwich       0.36      0.36      0.36       400
     sitting       0.65      0.57      0.61       495
        soup       0.60      0.70      0.65       361
      stairs       0.82      0.74      0.78       375
    standing       0.73      0.77      0.75       361
       teeth       0.82      0.88      0.85       369
      typing       0.82      0.67      0.74       520
     walking       0.84      0.92      0.88       371
     writing       0.75      0.69      0.71       449

    accuracy                           0.72      7279
   macro avg       0.73      0.73      0.72      7279
weighted avg       0.73      0.72      0.72      7279
```

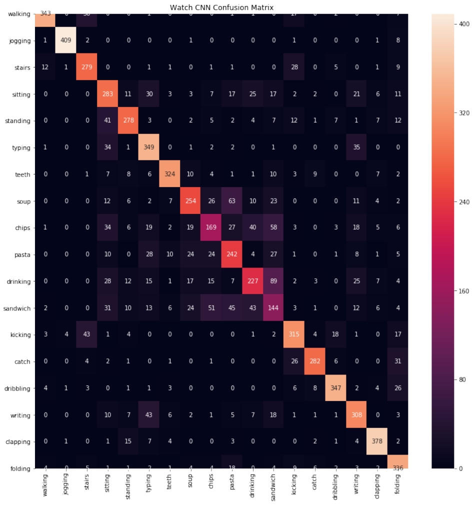
Figure 13: CNN results for Watch



Watch LSTM Confusion Matrix

	walking	jogging	stairs	sitting	standing	typing	teeth	soup	chips	pasta	drinking	sandwich	kicking	catch	dribbling	writing	clapping	folding
walking	685	2	32	1	0	0	0	1	0	1	0	1	22	0	1	0	2	20
jogging	7	771	9	1	1	0	0	0	0	3	0	0	6	5	0	0	1	3
stairs	22	5	581	4	1	1	6	0	0	0	1	5	72	22	6	1	0	22
sitting	3	3	1	616	23	24	7	8	32	26	25	17	1	1	3	13	0	10
standing	1	0	2	17	675	5	4	18	22	10	6	23	17	6	13	5	1	25
typing	1	0	0	43	0	630	0	9	17	24	9	10	1	1	0	50	1	3
teeth	1	0	4	1	5	0	783	16	21	1	4	17	1	1	0	0	14	10
soup	1	0	0	10	10	4	28	564	30	71	26	20	2	1	5	19	0	5
chips	1	0	1	36	19	16	13	33	469	53	50	108	2	4	1	6	0	13
pasta	2	0	0	15	7	10	0	102	63	542	9	41	0	3	0	10	1	14
drinking	0	0	0	23	7	7	6	30	50	15	598	71	2	2	0	25	1	13
sandwich	1	0	9	40	16	15	7	41	115	42	111	392	3	3	8	20	5	11
kicking	19	3	38	3	12	0	2	1	1	0	0	4	691	16	19	0	0	26
catch	2	0	8	2	7	1	0	1	2	1	4	3	21	700	12	1	0	29
dribbling	5	1	6	2	5	1	1	0	0	1	1	4	15	12	746	3	8	11
writing	0	0	0	12	2	25	12	5	7	14	21	12	2	1	2	740	0	2
clapping	0	1	0	1	30	0	3	0	3	4	1	1	1	2	2	0	729	2
folding	9	0	12	4	9	1	2	6	9	22	5	14	15	34	17	4	4	684

Figure 14: LSTM Confusion Matrix for Watch



Watch CNN Confusion Matrix

	walking	jogging	stairs	sitting	standing	typing	teeth	soup	chips	pasta	drinking	sandwich	kicking	catch	dribbling	writing	clapping	folding
walking	343	0	38	0	0	1	0	0	0	1	1	0	17	0	2	0	0	7
jogging	1	409	2	0	0	0	0	1	0	0	0	0	1	0	0	0	1	8
stairs	12	1	279	0	0	1	1	0	1	1	0	0	28	0	5	0	1	9
sitting	0	0	0	283	11	30	3	3	7	17	25	17	2	2	0	21	6	11
standing	0	0	0	41	278	3	0	2	5	2	4	7	12	1	7	1	7	12
typing	1	0	0	34	1	349	0	1	2	2	0	1	0	0	0	35	0	0
teeth	0	0	1	7	8	6	324	10	4	1	1	10	3	9	0	0	7	2
soup	0	0	0	12	6	2	7	254	26	63	10	23	0	0	0	11	4	2
chips	1	0	0	34	6	19	2	19	169	27	40	58	3	0	3	18	5	6
pasta	0	0	0	10	0	28	10	24	24	242	4	27	1	0	1	8	1	5
drinking	0	0	0	28	12	15	1	17	15	7	227	89	2	3	0	25	7	4
sandwich	2	0	0	31	10	13	6	24	51	45	43	144	3	1	0	12	6	4
kicking	3	4	43	1	4	0	0	0	0	0	1	2	315	4	18	1	0	17
catch	0	0	4	2	1	0	1	0	1	0	0	0	26	282	6	0	0	31
dribbling	4	1	3	0	1	1	3	0	0	0	0	0	6	8	347	2	4	26
writing	0	0	0	10	7	43	6	2	1	5	7	18	1	1	1	308	0	3
clapping	0	1	0	1	15	7	4	0	0	3	0	0	0	2	1	4	378	2
folding	4	0	5	1	1	2	1	4	4	18	0	4	9	6	2	3	2	336

Figure 15: CNN Confusion Matrix for Watch

7 Conclusion/Future Work

Weiss (6) intends to release an updated dataset which includes significantly more participants. Additionally, the set of tasks that each participant will complete will be larger. We intend to improve upon our results going forward on the expanded WISDM dataset that will be released later. Since the expanded dataset will also include more IMUs, we will investigate using different combinations of sensors to perform our learning task.

While our best performing model improved upon Weiss' results, we believe that with further fine-tuning of hyper parameters we can continue to improve the prediction accuracy of our approach. We will experiment with using a combination of the two kinds of networks commonly known as a Convolutional Recurrent Neural Network (**CRNN**). It is also worth noting that we have so far only used impersonal models for our classification task, however we will investigate creating unique models for each study participant as previous research has shown this approach to be more accurate.

8 Contributions

Derek McCreight and Susana Benavidez contributed equally in the paper, experiments, and code.

References

[1] Human activity recognition based on time series analysis using u-net. *DeepAI*, Sep 2018.

[2] A. R. AHad. Human activity recognition: Various paradigms. *Human activity recognition: Various paradigms - IEEE Conference Publication*, 2008.

[3] A. Bevilacqua. Convolutional neural networks for human activity ... Jun 2019.

[4] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. t. Hompel. Convolutional neural networks for human activity recognition using body-worn sensors. *Convolutional Neural Networks for human activity recognition using mobile sensors - IEEE Conference Publication*, May 2018.

[5] G. M. Weiss. Wisdm smartphone and smartwatch activity and biometrics dataset. *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set*, Sep 2019.

[6] G. M. Weiss, J. Timco, C. Gallagher, and K. Yoneda. Smartwatch-based activity recognition: A machine learning approach. *Smartwatch-based activity recognition: A machine learning approach - IEEE Conference Publication*, Apr 2016.

[7] Y. Zhao, G. Chevalier, and M. Gong. Deep residual bidir-lstm for human activity recognition ... *Arxiv*, 2019.