Report - Deep Activity Classifier

Pouya Kananian

September 22, 2019

Abstract

This report consists of the works done on a deep model for Human Activity Recognition (HAR) using hand accelerometer data, the different ideas that has been tested or can be implemented in future and a summary of some of the previous works done in this area and comparison of results with some of them. The report includes works done in summer 2019 at NUS university under supervision of Prof. Ooi Wei Tsang.

1 Introduction

In this work, the objective is to implement a deep model able to recognize different daily activities of a human wearing a smart watch. The model uses the data of the accelerometer sensor of the device for a small period of time to reach a deduction about the activities being done by the owner of the watch. The result of the classification is can be used to find out about the daily activities of a person to suggest user appropriate times for doing medical treatments and taking pills.

The dataset used in this work consists of 3 dimensions of accelerometer data. It is not produced in a synthetic envirment where one wears the sensor and is instructed to perform certain activities in a specific order. The dataset is captured during real daily activities of users and hence is expected to be more noisy than the synthetic datasets.

2 Problem Definition, Challenges and Project Phases

The first phase of the project includes implementation of a deep network for recognizing activities as a supervised learning task. 3 dimensions of accelerometer data are fed into the model and the network has to choose an activity among a certain number of activities. For each activity sufficient data is fed to the network in the training phase and in the test phase the model should be able to recognize each activity with a good accuracy.

The most outstanding challenges of implementing such a model are listed below:

- Working with limited noisy wrist data: Many of the previous works use a lot of sensors in various parts of body [3] [8] [11] [12] but our dataset

is limited to wrist accelerometer sensor which is quite noisy. It is worth mentioning that different kind of noise could be present in our dataset:

  - Sensor noise: Recorded time series of sensors capture movements of users with some error. This noise is present in all the captured series but this lack of accuracy in capturnig movements doesn't have a big effect on model's performance as long as the sampling rate of the sensor is large enough.
  - Noisy labels: Doing a certain activiy may coincide with another activity in daily life. As an example, while walking, a user may shake her hand to greet a friend. The recorded time series for this action is quite different from normal data of walking and yet such a series may be present in the dataset labeled as walking. Various activities coinciding with walking can generate time series unsimilar to normal data. These series could be interpreted as wrongly labeled but such time series are present in daily life and hence cannot be removed from dataset. We will suggest a way for handling them further in the text.

- Multiple distributions in acclerometer data for a certain activity: Different people do an activity in different ways. They might have different speeds and gestures. Even for one person, there are various gestures that could occur during doing a certain task. Some of the gestures may occur seldomly while some of them are present all over the dataset. This variety and the fact that the same amount of data is not available for all gestures make the classification task difficult.

On the second phase after having a model able to classify human activities with a good accuracy, there are several ways for improving our model's performance in real life. Some of the ideas are listed below:

- Adapting to one user: As mentioned before, gestures of various users could be quite different with each other. If we could propose a model which is able to update and adapt itself to a certain user during working, using reasonable computing power and energy, we would be able to satisfy each user with a much better recognition accuracy. In order to achieve this, a simple idea is to update some limited weights of the model for each user. Generally various ideas of Online and Active learning could be used for personalizing the network for one person.

- Detecting new activities that hasn't been seen before: The activities that the model is trained on them are limited. In real daily life a lot of activities could occur that are unknown to the model. If the model has the ability to detect new activities that hasn't been seen before, by asking the type of activity from the user, it can add the new activity to the set of known ones and practice to recognize it during working. Online and Active learning techniques could be used for designing such a model.

3 Previous Deep Learning Works in This Area

Various deep learning approaches have been tested for Human Activity Recognition. [1] Those that are most related to our work include works that use

Recurrent Neural Networks [10] [11] [12], those that use Convolutional Networks [2] [3] [4] [5] [6] [7] and works that use a combination of convolutional and recurrent networks [8] [9].

4 Deep Network Development, Results, Comparisions and Ideas

4.1 Datasets

Before going through the details of the development of the deep network, we take a look at the detasets that are used in this work.

4.1.1 Our Dataset

- Our dataset is collected using an accelerometer sensor placed on the dominant hand of the user. The collected data consists of 3 time series related to Cartesian axises of space. The dataset is not completed yet and hence is changing and growing but at the time this report is wrote, it consists of data of 2 users and 18 different daily activities.
- **Size:** For each recorded activity we divide the recorded time series into smaller segments. When the length of these segments are set to 10 seconds and there is no overlap between segments, there are 2777 segments in the dataset. Unfortunately not enough data is available for all the activities. The activities that we have enough data for, are walking, commute in bus, working on laptop and sleeping.

4.1.2 Chest Accelerometer Dataset

- This dataset [13] is collected from an accelerometer sensor placed on chests of the user. For each activity 3 time series are available related to 3 Cartesian axises of space. The dataset is collected from 15 users and consists of 8 different activities.
- **Size:** If we create segments of 10 second length from the recorded series of different activities there would be 5435 different series available which don't overlap with each other. There is enough data for 4 activities (Working at Computer, Standing, Walking and Talking while standing) but for the 3 other activities the data doesn't seem to be enough for a deep model.

4.1.3 Wharf Dataset

- This dataset [14] is a collection of tri-axial accelerometer data placed on a single wrist-worn device. The dataset is collected from 17 users and consists of 14 different activities.
- **Size:** This dataset is relatively small and therefore is only used in some of the initial tests when the deep model wasn't as complex as it's current state. The dataset contains 1703 samples of 10s length and that is pretty small considering the large number of activities and users in this dataset.

4.1.4 MHealth Dataset

- This dataset [15] includes data of various accelerometer and gyroscope sensors installed on different parts of body. The data of wrist accelerometer is used for testing our model.
- **Size:** This dataset contains 3503 samples with 10 seconds length. Data is collected by 10 users on 13 different activities.

4.1.5 PAMAP2 Dataset

- This dataset [16] includes data of various sensors installed on different parts of body. This dataset contains 52 different time series for each activity. The data of wrist accelerometer is used for testing our model.
- **Size:** This dataset is the largest and is used for testing the performance of the model when enough large data is available. This dataset is used by some other deep learning works and this fact enables us to compare the performance of our network with others. It contains 8075 non-overlapping samples with 10 seconds length. Data is collected by 9 users on 13 different activities.

4.2 Network Evolution

This section includes a summary of the models implemented, their results on different datasets and the steps taken to make them better.

4.2.1 Simple RNN

The first network implemented was a simple recurrent neural network with LSTM cells. Input of a network was a 3 dimensional time series and it's length was 360. To be more precise the size of the input that is fed to the model in each batch was $batch\_size * 360 * 3$. The sampling rate of the accelerometer is 36 per second hence a time series with length 360 represents 10 seconds of data. The hidden state of the last cell was passed to a fully connected layer in charge for classifying the activity. The number of outputs of the fully connected layer was equal to the number of activities and a one-hot vector at the output shows the label of the classified activity. Models details are listed below:

- network schematic: $input \rightarrow rnn \rightarrow fc \rightarrow output$
- hidden neurons of LSTM cells: 32
- fully connected layers: 3

Model's perfomance:

- test accuracy on our dataset: 0.57

4.2.2 RNN + Embedding Layer

In order to enhance the performance of the previous model we added to the complexity of it by adding an embedding layer and a pooling layer.

In this model the input goes through an embedding layer before entering the RNN. If we show the input matrix of the network with X, the embedded output could be shown by Y = MX where M is the embedding weights which are learned in the same way that all the networks weights get learned using backpropagation.

The hidden states of neurons of the cells are the inputs of the pooling layer. The pooling layer summarizes the output of all the neurons and is implemented similar to the pooling layer in the deep network suggested by Gou et al. [17].

The details of this network is as follows:

- network schematic: $input \rightarrow embedding\,layer \rightarrow rnn \rightarrow pooling\,layer \rightarrow fc \rightarrow output$
- input len: 10 seconds
- hidden neurons of LSTM cells: 32
- pooling layer: mean pooling, last pooling, max pooling
- fully connected layers: 3

Model's performance:

- test accuracy on our dataset: 0.78

4.2.3 Convolutional LSTM Network

In order to enhance the performance of the previous model we tested a convolutional layer instead of the embedding layer that used a matrix product. Convolutional LSTM networks has previously been used for activity recognition [8] [9] and had good performances.

Various tunings of this model has been tested on different datasets. The following hyperparameters had the best performance among the tested numbers.

- Model's details:
  - network schematic: $input \rightarrow convolutional\,layer \rightarrow rnn \rightarrow pooling\,layer \rightarrow fc \rightarrow output$
  - input len: 10 seconds
  - hidden neurons of LSTM cells: 32
  - CNN filter size: (6, 3), number of filters: 5
  - pooling layer: mean pooling, last pooling, max pooling
  - fully connected layers: 3
- Model's perfomace:
  - test accuracy on our dataset: 0.79
  - test, train accuracy on Chest Accelerometer dataset: 0.56, 0.71

- test, train accuracy on normalized Chest Accelerometer dataset: 0.60, 0.72
- test, train accuracy on normalized Chest Accelerometer dataset when only data of one user is used for training and testing: 0.79, 0.98
- test accuracy on normalized Wharf dataset: 0.55

4.2.4 Convolutional LSTM Network + TimeDistributed Layer

In order to enhance the performance of the previous model we further added to the complexity of the network by using a TimeDistributed layer. This layer adds a fully connected layer to the output of each cell of RNN. The weights of these fully connected layers is shared throughout the network.

Details of one of the best tunings of this model and it's results are listed below:

- Model's details:
  - network schematic: $input \rightarrow convolutional\,layer \rightarrow rnn \rightarrow time\,distributed\,layer \rightarrow pooling\,layer \rightarrow fc \rightarrow output$
  - input len: 10 seconds
  - hidden neurons of LSTM cells: 64
  - CNN filter size: (6, 3), number of filters: 5
  - hidden neurons of TimeDistributed layer: 64
  - pooling layer: mean pooling, last pooling, max pooling
  - fully connected layers: 3
- Model's perfomace:
  - test accuracy on our dataset: 0.87
  - test, train accuracy on normalized Wharf dataset: 0.52, 0.98

4.2.5 Adding Dropout and other minor changes

In order to enhance the performance of the previous model some minor changes were tested:

- *Overlapping input segments:* Previously the input segments used had no overlap with each other. We tested 20 percent of overlap between the inputs but doing this only helped getting a better train accuracy due to overfitting and didn't enhance the test accuracy and hence we changed back to non-overlapping segments.
- *Applying cnn filters to each dimension of the input separately:* We changed the shape of the cnn filters from (6, 3) to (15, 1). In this way filters are applyed to each axis of input separately. Applying the filters to each dimension had a better result. We also tested having 3 separate RNNs for each axis of the input which had the same result as using filters with size (m, 1). Since the computational power of using 3 RNNs is greater than using filters with the right shape, we dropped the idea of 3 separate RNNs.

- *Dropout:* We applyed dropout to different layers of network. Applying dropout to the convolutional layer will decrease both train and test accuracy and does not have a good effect whereas applying dropout to fully connected layers will not decrease test accuracy and helped decreasing overfitting.
- *Bidirectional RNN:* We tested a bidirectional RNN instead of the simple one directional one that was used previously. It slightly improved the performance of the network but it adds to the complexity of network and causes the model to learn the noise and overfit to train data.
- *Feeding the network with lesser data:* Instead of feeding the network with 10s of data shorter time series could be fed to network. Although shorter time series contain lesser data, our network had a better performance when was fed with 5s or 2.5s of data. It will be mentioned further that this is not true for a more complex network. A model complex enough would be able to use the extra information in a 10s time series.

The attributes of one of the networks that had the best performance applying the above-mentioned changes could be find below:

- Model's details:
  - network schematic: $input \rightarrow convolutional\,layer \rightarrow rnn \rightarrow time\,distributed\,layer \rightarrow pooling\,layer \rightarrow fc \rightarrow output$
  - input len: 2.5 seconds
  - BiDirectional RNN was used
  - hidden neurons of LSTM cells: 64
  - CNN filter size: (30, 1), number of filters: 5
  - hidden neurons of TimeDistributed layer: 64
  - pooling layer: mean pooling, last pooling, max pooling
  - fully connected layers: 3
  - Dropout @ fully connected layer, keep probability: 0.9
- Model's performance:
  - test, train accuracy on our dataset: 0.857, 0.97
  - test, train accuracy on our dataset when only the activities with enough data was used for training and testing: 0.93, 1
  - test, train accuracy on normalized MHealth dataset: 0.72, 0.98
  - test, train accuracy on Chest Accelerometer data when 4 of the activities that had enough data was used: 0.65, 0.85
  - test, train accuray on PAMAP2 dataset: 0.63, 0.93

As can be seen in the results above, the model's performance on our dataset is better than other ones. The number of users in our dataset is only 2 whereas in other datasets it is more. Model's performance on other datasets when smaller number of users where used where also quite better. Gestures of various users are different and as the number of users increase learning the multi distributed input becomes a harder task. As mentioned further we added to complexity of the model, hoping that it would be able to learn all the distributions.

4.2.6 Conv LSTM Network with 3 RNNs

In order to enhance the performance of the previous model we tried adding to the complexity of model. Adding to complexity of the network will add to the energy used by the model in real life application and decrease it's throughput but it's still feasible to use more complex networks in smart phones. [9] For adding to the complexity of network we could either add to the layers of the recurrent network or the convolutional network.

Adding to the complexity of the recurrent network was tested first. Inspired from a work done by Inoue et al. [10] we used three layers of RNN instead of one. The TimeDistributed layer of the previous networks was dropped to test the effect of 3 RNNs without having this layer.

PAMAP2 dataset is the largest dataset available and due to the high complexity of this network, only this dataset is used for testing.

- Model's details:
  - network schematic: $input \rightarrow convolutional\,layer \rightarrow 3\,layers\,of\,rnn \rightarrow pooling\,layer \rightarrow fc \rightarrow output$
  - input len: 2.5 seconds
  - One directional RNN was used
  - hidden neurons of LSTM cells: 64
  - CNN filter size: (30, 1), number of filters: 5
  - pooling layer: mean pooling, last pooling, max pooling
  - fully connected layers: 3
  - Dropout @ fully connected layer, keep probability: 0.9
- Model's performance:
  - test, train accuracy on PAMAP2 dataset: 0.64, 0.96

4.2.7 Complex Conv LSTM Network

In order to enhance the performance of the previous model we tested adding to the complexity of convolutional layer. We used one layer of one directional RNN in this network to avoid too much complexity. The details of one of the models with good hyperparameters and results is listed below.

- Model's details:
  - network schematic: $input \rightarrow 3\,layers\,of\,convolutional\,layer \rightarrow rnn \rightarrow pooling\,layer \rightarrow fc \rightarrow output$
  - input len: 2.5 seconds
  - One directional RNN was used
  - hidden neurons of LSTM cells: 64
  - CNN filter size: (10, 3), (6,3), (3,3) , number of filters: 20, 20, 10
  - TimeDistributed layer was not used
  - pooling layer: mean pooling, last pooling, max pooling

- fully connected layers: 3
- Dropout @ fully connected layer, keep probability: 0.9

- Model's performance:
  - test, train accuracy on PAMAP2 dataset: 0.72, 0.97

Model 4.2.7.1: Personalized Complex Conv LSTM Network: We tested personalizing model 4.2.7 on one user. For doing so, we extracted data of one of the users from the dataset, trained the network on all the users except the picked one for 100 epochs and then trained it for 50 epochs on data of the single picked user. This model ,which is inferred in github code as model 4.2.7.1, could improve the accuracy for the picked user and could reach 75 percent test accuracy.

Testing the effect of Sampling rate: We tested the effect of reducing sampling rate of input series on the performance of the model. PAMAP2 dataset was used. When we changed the rate to half of it's original value, the test accuracy of model 4.2.7 decreased only 2 percent but the accuracy of personalized model (4.2.7.1) dropped significantly to 50 percent.

4.2.8 Collaboration of two Complex Conv LSTM Models in a Co-Teaching network

An analysis of model 4.2.7 could be find in part 4.3. In order to deal with it's overfitting problem, we tried using two such models in a co-teaching[20] network. The two network help each other ignore noisy labels during training. Studies show deep networks first capture useful information and features of the dataset and then overfit to noise[20]. By ignoring noisy labels during training, networks should be able to reduce overfitting.

When we tested a co-teaching network on PAMAP2 dataset we got 0.68 test and train accuracy. While overfitting was no longer present, test accuracy didn't got better.

4.3 Analysis of Complex Conv LSTM Network and Future works

Model 4.2.7 has the best performance among the introduced models. Despite of its complexity it is unable to achieve a good test accuracy on the PAMAP2 dataset.

Other deep learning works that has been tested on PAMAP2 dataset [3] [11] [12] are able to reach test accuracies better than 95 percent but as explained in the dataset section, PAMAP2 dataset includes data of many sensors in different parts of body (the complete dataset has 52 dimensions) and those works use all of the attributes of dataset. It is a much harder task to classify the activities using solely the hand accelerometer data. To confirm this we tested our model with all the dimensions of the PAMAP2 and got 93 percent test accuracy. This was without doing any tuning on the hyperparameters of the model and the model should be able to compete with other works using better hyperparameters.

There are some works that get good accuracies using hand accelerometer data but their datasets are not tested. Testing their datasets is among TODO

works. It is also needed to test their model on PAMAP2 dataset to see their performance.

4.3.1 Model's Problems

- **Overfitting:** As mentioned in section 2 hand accelerometer data could be quite noisy and deep models are vulnerable to noise and overfititing. Figure 1 and 2 show loss and accuracy plots of train and validation data during data. It can be seen that the model starts overfitting and learning the noise after a while. While train accuracy continues to get better, validation accuracy does not improve. We also observe that validation loss becomes increasing after a while.

- **Dealing with multi distributed data:** As opposed to the previous problem, the problem mentioned in this section is not a confirmed problem of the model. We only guess that the model is not able to deal with our multi distributed data well. Model 4.2.7 is able to reach 72 percent test accuracy on PAMAP2 dataset and one of the reasons that it cannot reach a better test accuracy might be multi distributed data.

  Having many users and the fact that various gestures might be observed during doing a certain activity makes the input data multi distributed and learning all the distributions is a hard task. Figure 3 shows x axis of data collected for ironing and two dissimilar time series could be seen. Similar to ironing, all the activities include of various distributions. For some of the distributions and gestures more data is present in the dataset and that might make the model ignore other distributions. Deep models should be able to capture all the different distributions of input data due to their high complexity but they are only able to do so if fed with sufficient samples of each distribution. We will try to investigate whether this problem is present in our model or not further in the text.

4.3.2 Analysis of model's problems: Clustering input data

In order to reduce the problems mentioned in previous sections we needed to find their source.

Overfitting occurs when the model learns noise instead of useful information. We needed to confirm that our data is noisy and find out if there are useful information that are being ingnord by the model in order to learn the noise instead.

We also needed to confirm that the model is sufffering from multi distributed input. In order to do so, we clustered input series of different clusters and checked model's accuracy on different clusters of each activity. We used a hierarchical clustering based on DTW distance between time series. Dendograms was used to determine a reasonable number for clusters of each activity. Dendogram plots also help us diagnose noisy data from clusters belonging to different gestures of an activity.

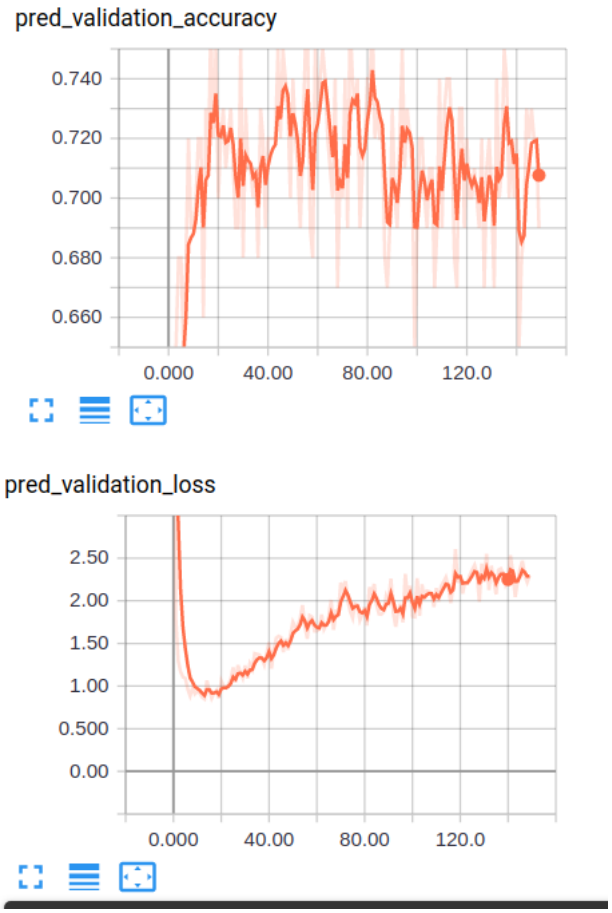

Figure 1: Validation accuracy and loss, complex conv LSTM model when fed by hand accelerometer data

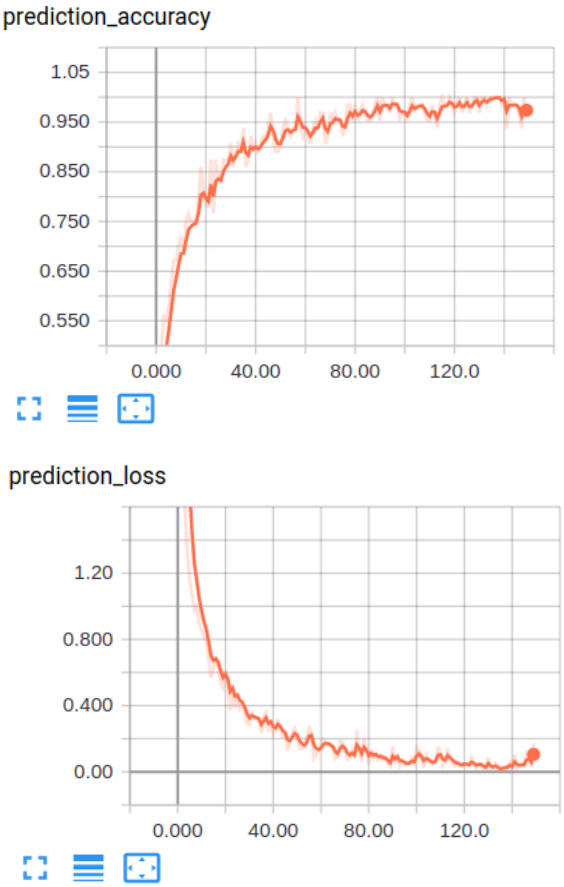

Figure 2: Train accuracy and loss, complex conv LSTM model when fed by hand accelerometer data

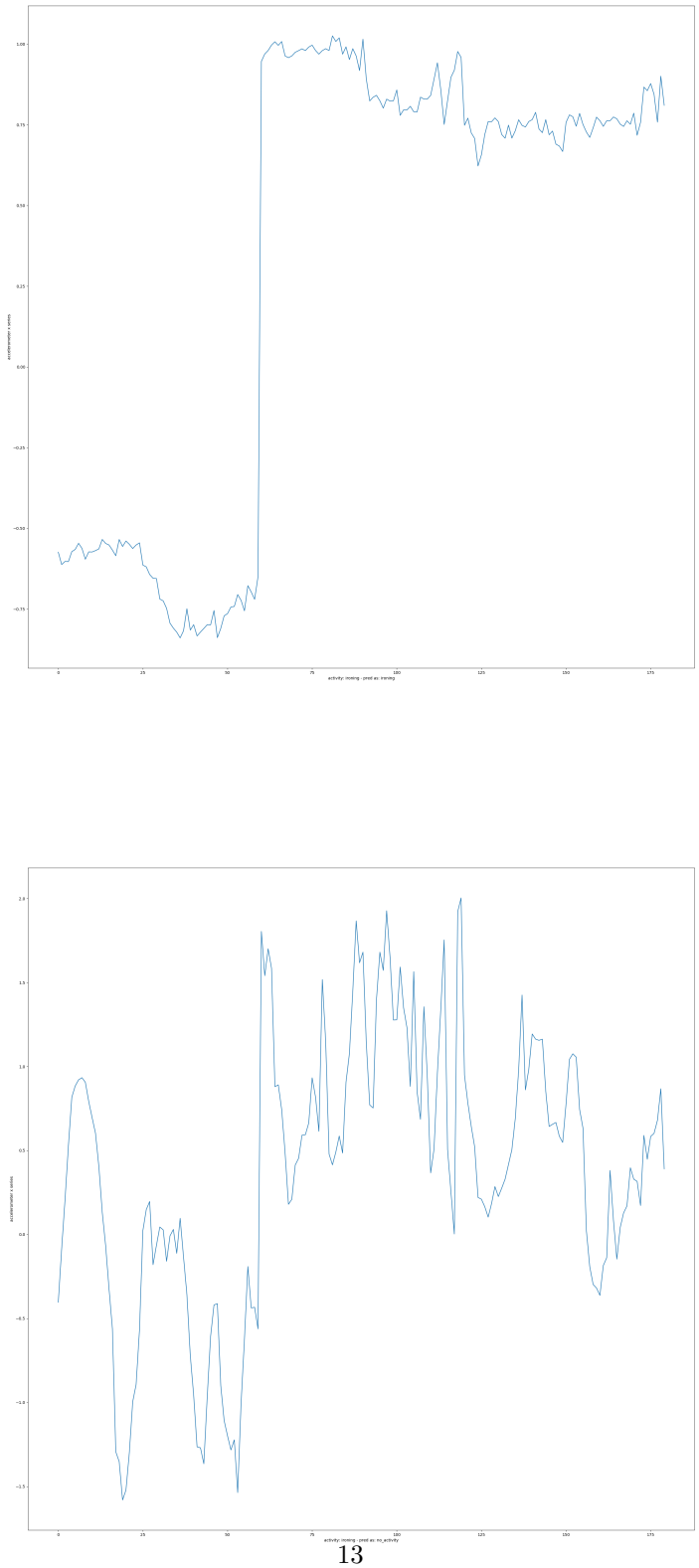

Figure 3: X axis of accelerometer data for ironing shows two dissimilar time series belonging to one activity

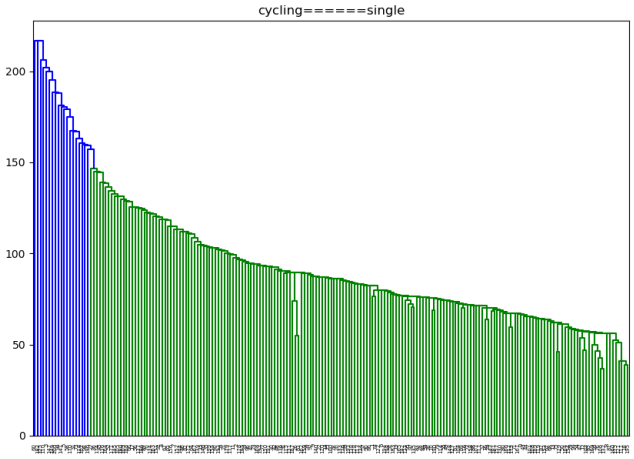

Figure 4: Dendogram of cycling data, PAMAP2 dataset

Figure 4, 5 and 6 show dendograms of some of the activities of PAMAP2 dataset. As could be interpreted from the diagrams, cycling doesn't have more than one distribuion and there are some noisy samples which could be spotted by blue lines in the dendogram. For nordic walking and running multiple distributions could be spotted in the diagrams. We clustered cycling time series into two clusters and each of the other mentioned activities into 3 clusters. Analysing the performance of a trained 4.2.7 model on these clustered time series showed that the model performs well on one of cycling clusters and on two clusters of nording walking and two clusters of running. The model has a bad performance on one of the clusters of each activity (about 40 percent accuracy was observed). The clusters that the model has problem dealing with belong to noisy data. For running and nordic walking there are two clusters that we get very good accuracies on (more than 90 percent). Despite the fact that the size of these clusters are different and there are much more samples for some of them, the smaller clusters are not being ignored by the model and hence we conclude that the model is actually able to cope well with multi distributed data. All the distributions are being considered by the model and it's only problem is noise.

4.3.3 Suggestions for improving the model

Based on the analysis in previous section we concluded that noisy data is causing the unsatisfactory test accuracy of the model. In this section we provide some suggestions for improving the performance of our model in daily life:

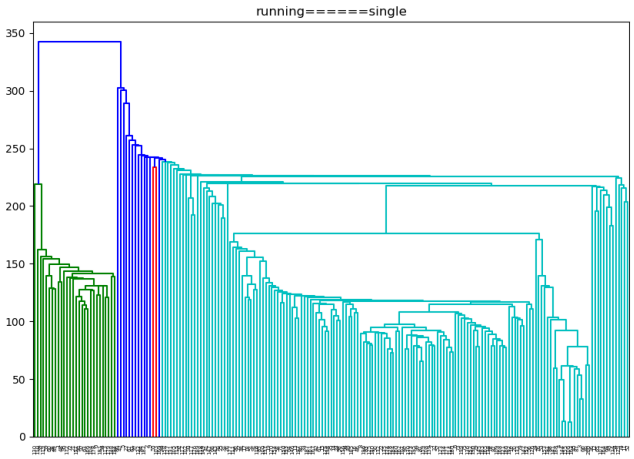

Figure 5: Dendogram of running data, PAMAP2 dataset

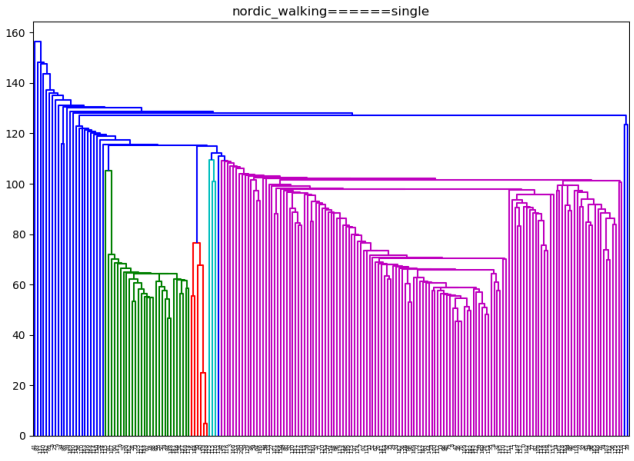

Figure 6: Dendogram of nordic walking data, PAMAP2 dataset

- **Active Online Learning:** Model's performance depends on the data it is being fed with. Short time series of length 2.5 seconds are being used as it's inputs and in daily life, many of these short series may not be informative enough. While doing a simple daily activities like walking a lot of unrelevant gestures may be captured by movements sensors. As an exmaple, if the user shakes her hand while walking or gets frightened by a monkey in the middle of the walk, a time series could be recieved by the model which is not similar to what has been seen before and could be recognized as another activity while the user may still continue to walk after those events. In order to prevent such errors, online and active learning could be used. Different ideas such as the following methods could be tested:
  - A learning method could be used for finding out whether an input time series in informative enough or not. If it is not good enough for deciding about the activity the model should ask for more data instead of making a decision. In other words model could be trained to find out which inputs need to be ignored.
  - Model could be personalized for a certain user while being used. Some weights of the model could get updated while working based on the recieved inputs from user and asking questions about the type of activities. By doing so, not only the model would be able to perform better on each user, but also it would be able to recognize and gather data for new activities that are frequent in user's life and improve it's performance on diagnosing whether a time series is informative enough or not.

5 Future works

In this section we list some of the works that need to be done in future:

- As mentioned in part 4.3, comparnig to other works on PAMAP2 dataset we only use hand accelerometer data and therefore, our accuracy is lower than other works. We tested using all the attributions of the dataset and our model could compete with the previous works then. It is useful to test previous works on hand accelerometer data of dataset to confirm that those models struggle to get a good result with limited features same as our model. In this way it could be confirmed that there is no problem with our model and it is the difficulty of the task that leads to low accuracy.
- In section 4.3.2 we claimed that noisy data is the reason of unideal test accuracy. To confirm this we can remove noisy data from the dataset, train and test the model on clean data, and check if the model is able to achieve a good test accuracy in this situation.
- We can test the model with different hyperparametes and try to enhance it's performance by playing with hyperparameters. Different configs has been tested already but there still might be a config that hasn't been tested and can improve model's performance.

- Implementing an online active learning method which we discussed about in section 4.3.3 is another important future work.

References

[1] Wang, Jindong, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. "Deep learning for sensor-based activity recognition: A survey." Pattern Recognition Letters 119 (2019): 3-11.

[2] Zeng, Ming, Le T. Nguyen, Bo Yu, Ole J. Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. "Convolutional neural networks for human activity recognition using mobile sensors." In 6th International Conference on Mobile Computing, Applications and Services, pp. 197-205. IEEE, 2014.

[3] Hammerla, Nils Y., Shane Halloran, and Thomas Plötz. "Deep, convolutional, and recurrent models for human activity recognition using wearables." arXiv preprint arXiv:1604.08880 (2016).

[4] Sathyanarayana, Aarti, Shafiq Joty, Luis Fernandez-Luque, Ferda Ofli, Jaideep Srivastava, Ahmed Elmagarmid, Shahrad Taheri, and Teresa Arora. "Impact of physical activity on sleep: A deep learning based exploration." arXiv preprint arXiv:1607.07034 (2016).

[5] Pourbabaee, Bahareh, Matthew Howe-Patterson, Eric Reiher, and Frederic Benard. "Deep convolutional neural network for ECG-based human identification." CMBES Proceedings 41 (2018).

[6] Jiang, Wenchao, and Zhaozheng Yin. "Human activity recognition using wearable sensors by deep convolutional neural networks." In Proceedings of the 23rd ACM international conference on Multimedia, pp. 1307-1310. Acm, 2015.

[7] Singh, Monit Shah, Vinaychandran Pondenkandath, Bo Zhou, Paul Lukowicz, and Marcus Liwickit. "Transforming sensor data to the image domain for deep learning—An application to footstep detection." In 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2665-2672. IEEE, 2017.

[8] Ordóñez, Francisco, and Daniel Roggen. "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition." Sensors 16, no. 1 (2016): 115.

[9] Yao, Shuochao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. "Deepsense: A unified deep learning framework for time-series mobile sensing data processing." In Proceedings of the 26th International Conference on World Wide Web, pp. 351-360. International World Wide Web Conferences Steering Committee, 2017.

[10] Inoue, Masaya, Sozo Inoue, and Takeshi Nishida. "Deep recurrent neural network for mobile human activity recognition with high throughput." Artificial Life and Robotics 23, no. 2 (2018): 173-185.

[11] Edel, Marcus, and Enrico Köppe. "Binarized-blstm-rnn based human activity recognition." In 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-7. IEEE, 2016.

[12] Guan, Yu, and Thomas Plötz. "Ensembles of deep lstm learners for activity recognition using wearables." Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1, no. 2 (2017): 11.

[13] Casale, Pierluigi, Oriol Pujol, and Petia Radeva. "Human activity recognition from accelerometer data using a wearable device." In Iberian Conference on Pattern Recognition and Image Analysis, pp. 289-296. Springer, Berlin, Heidelberg, 2011.

[14] Bruno, Barbara, Fulvio Mastrogiovanni, and Antonio Sgorbissa. "A public domain dataset for ADL recognition using wrist-placed accelerometers." In The 23rd IEEE International Symposium on Robot and Human Interactive Communication, pp. 738-743. IEEE, 2014.

[15] Banos, Oresti, Rafael Garcia, Juan A. Holgado-Terriza, Miguel Damas, Hector Pomares, Ignacio Rojas, Alejandro Saez, and Claudia Villalonga. "mHealthDroid: a novel framework for agile development of mobile health applications." In International workshop on ambient assisted living, pp. 91-98. Springer, Cham, 2014.

[16] Reiss, Attila, and Didier Stricker. "Creating and benchmarking a new dataset for physical activity monitoring." In Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments, p. 40. ACM, 2012.

[17] Gou, Chengcheng, Huawei Shen, Pan Du, Dayong Wu, Yue Liu, and Xueqi Cheng. "Learning sequential features for cascade outbreak prediction." Knowledge and Information Systems 57, no. 3 (2018): 721-739.

[18] Xiao, Chaowei, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. "Generating adversarial examples with adversarial networks." arXiv preprint arXiv:1801.02610 (2018).

[19] Shen, Shiwei, Guoqing Jin, Ke Gao, and Yongdong Zhang. "Apegan: Adversarial perturbation elimination with gan." arXiv preprint arXiv:1707.05474 (2017).

[20] Han, Bo, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. "Co-teaching: Robust training of deep neural networks with extremely noisy labels." In Advances in neural information processing systems, pp. 8527-8537. 2018.