MyBreathingHeart: Preliminary Experiments

Andrea Ceni*†[1], Claudio Gallicchio‡[1], and Vincenzo Gervasi§[1]

[1]Department of Computer Science, University of Pisa, Italy

September 1, 2022

Abstract

In this brief report we highlight some of the preliminary results obtained on the early data collected in the initial phase of the MyBreathingHearth (MBH) project.

1 Structure of MBH data

Data was collected between the 29$^{\text{th}}$ of July and the 2$^{\text{nd}}$ of August 2022 on a total of 51 subjects. For each individual, five different recording settings were implemented, that will be referred to as *scenes*. In all scenes the subject is lying supine with the smartphone screen facing upward, and the bottom of the smartphone (where the microphone is usually located) oriented towards the left part of the subject's body:

- Scene 1. Smartphone placed on the right chest.
- Scene 2. Smartphone placed on the central chest.
- Scene 3. Smartphone placed on the left chest.
- Scene 4. Smartphone placed on the abdomen slightly above umbilicus.
- Scene 5. Smartphone placed on the abdomen slightly below umbilicus.

For each scene of each subject, a number of sensor time series have been recorded. Namely,

- 6 inertial sensor channels: 3 axial accelerometer time series signals (a_x, a_y, a_z), and 3 axial gyroscopic time series signals (g_x, g_y, g_z).
- 2 pulsoximeter signals: heart rate beats per minute (hr), and saturation of peripheral oxygen (spo2).
- 1 stereo audio recording.

*andrea.ceni@di.unipi.it
†Corresponding author
‡claudio.gallicchio@unipi.it
§vincenzo.gervasi@unipi.it

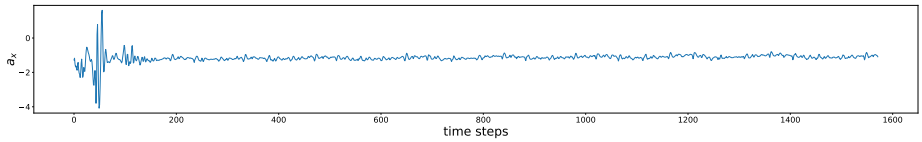

Figure 1: Example of an inertial time series presenting a transient behaviour.

In this preliminary report we will focus only on non-auditory data, and mainly on inertial time series data.

All data were recorded for a period of about 30 seconds approximately simultaneously for each sensor (inertial, pulsoximeter, audio); inertial data at frequencies ranging in 50–60 Hz [1], while pulsoximeter data at about half the frequency of inertial data. As a consequence, inertial time series have time steps ranging in 1565–1577 time steps, while pulsoximeter time series ranging in 743–748 time steps.

It is worth to mention that all the collected recordings come from presumably healthy people, apart from 2 subjects which were labelled as "Allergy", 1 subject labelled as "Asthma", and 1 subject labelled as "Bronchitis". As a consequence, at this stage of the project we will focus mainly on data cleaning, visualisation, and attempt to detect anomalies, or cluster data in an unsupervised way.

2 Preprocessing

2.1 Washout from transients

As a very first step, we need to clean the data. In fact, the vast majority of time series present a transient behaviour in the beginning of the recordings. In the plot in Fig. 1, it can be seen an example of transient dynamics in an a_x time series.

We want to wash out these transient dynamics since they do not represent cardiorespiratory patterns of the specific subject which is recording. Contrariwise, there is no need to do it for the pulsoximeter data, since there is no corruption due to movements of the subject in the beginning of the recording session. Moreover, in this phase we do not deal with audio data, but presumably, they will need to be cleaned as well.

2.1.1 Window estimation

For the purpose of estimating the transient length, we devised the following procedure.

- For each inertial channel of each recording, compute the moving standard deviation (*movSTD*) time series with a window length of 50 time steps. Several window lengths have been explored, and 50 gave reasonably good results.
- Compute the mean (m) and standard deviation (s) of each *movSTD* time series.

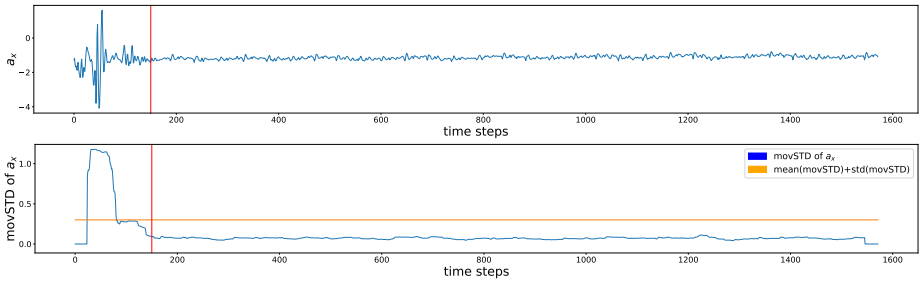

Figure 2: **Top:** the red vertical line shows the result of the procedure for the estimation of the end of the transient dynamics. **Bottom:** the *movSTD* time series derived from the a_x time series of the top plot. In orange the threshold $m + s$, where m is the mean value of *movSTD*, and s the standard deviation of *movSTD*.

- Compute the first time step (T) of *movSTD* such that all the steps from $movSTD[T]$ to $movSTD[T + 50]$ have values below $m + s$.
- Estimate the end of the transient dynamics at the time step $T + 50$, i.e. adding the length of the window used for the computation of *movSTD*.

Fig. 2 shows an example of estimation of the end of the transient dynamics of an inertial time series. In the bottom plot it is shown the *movSTD* corresponding to the time series of the above plot. When the *movSTD* intersect the orange horizontal line, and stays under it for at least 50 time steps, then it is an indicator that the transient dynamics is finished.

2.1.2 Common transient length

Additionally, since we might need to handle time series of all the same length, it is convenient to set a common window to cut off most of the transient dynamics. The maximum transient among all inertial time series resulted to be 483 time steps. However, it seems unfair to cut off such a large transient window. In fact, the very maximum estimated transient length is quite an outlier if compared with all the other estimated transient lengths, as can be seen from the histogram in Fig. 3.

For this reason, we selected 340 as the number of transient time steps to get rid of, while excluding from the dataset the 4 outlier recordings with more than 340 time steps of transients; namely the (subject, scene) pairs of (6,4), (17,1), (17,2), (40,4). Now, the minimum length among all the inertial time series is 1565 time steps. Therefore, the inertial dataset we will use is formed by the last 1233 time steps of the originally recorded inertial time series. We opted for the last 1233 time steps, instead of the first 1233 time step, in order to avoid transients dynamics which might not have been properly washed out.

2.2 Intertial and pulsoximeter alignment

Time series of hr and spo2 coming from the pulsoximeter are recorded with a different frequency of that coming from inertial sensors. In Fig. 4 an a_y inertial

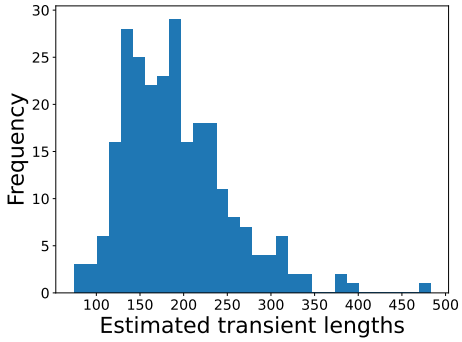

Figure 3: Histogram of estimated transient lengths, for all inertial time series.

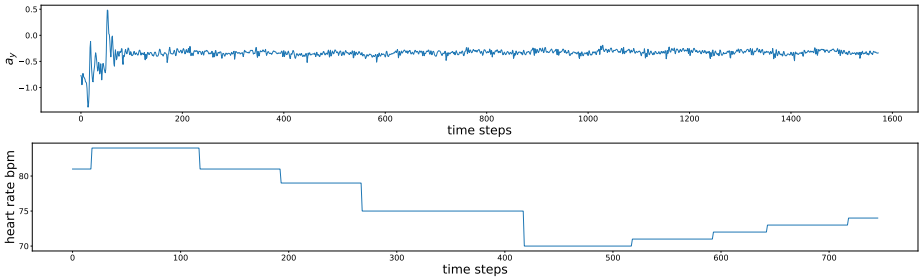

Figure 4: Comparison example between inertial and pulsoximeter time series, prior to the wash out of transients.

time series and its corresponding hr time series are plotted.

In order to transform the pulsoximeter time series to have the same length of the inertial time series we implemented the following procedure[1].

- Compute the minimum length of the original inertial time series, prior to wash out transients. This resulted to be 1565 time steps.
- Compute the minimum length of the pulsoximeter time series. This resulted to be 743 time steps.
- Compute the converting factor $F = 1565/743 \approx 2.1$. This means that the i-th value of the hr time series corresponds roughly to the window of time steps between $\lfloor i * F \rfloor$ and $\lfloor (i+1) * F \rfloor$ of the inertial time series, where $\lfloor x \rfloor$ denotes the inferior whole part of x.
- Thus, enlarge the hr time series by the factor F, reproducing the same hr[i] values at time steps from $\lfloor i * F \rfloor$ to $\lfloor (i+1) * F \rfloor - 1$.
- Finally, extract the last 1261 time steps in order to be coherent with the inertial time series washed out from the transient.

This procedure highlighted that 4 pulsoximeter recordings failed, since their hr and spo2 time series resulted of zero length; specifically those relative to (subject, scene) pairs of $(13, 3), (26, 3), (31, 5), (42, 1)$.

[1]We intend to implement a similar approach for aligning the audio data as well.

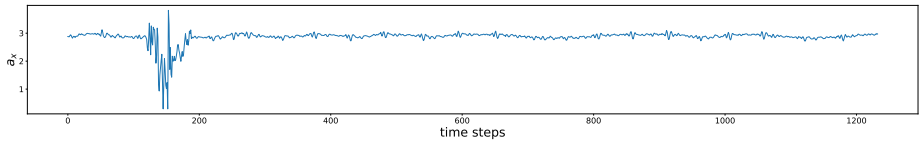

Figure 5: Example of a particularly evident corrupted recording in the channel a_x. These corruptions occur outside of the unwanted transient dynamics.

2.3 Corrupted recordings

Other than transient dynamics, the inertial data present occasionally sudden bursts of activity, probably due to movements while recording. In the plot below in fig. 5 it is shown one of these bursts in the a_x time series.

Those kind of corruptions might be detrimental for the model to train. We do not want the machine to learn these kinds of dynamics which are not inherent to the "typical" cardiorespiratory patterns. Therefore, we need to devise a procedure to seek out those behaviours. As a first attempt, we simply aim to recognise these anomalous time series presenting large in amplitude excursions, and get rid of them.

2.3.1 Inertial data

In order to detect the inertial data recordings corrupted by sudden moves, we first skimmed inertial time series according to the following strategy.

- Normalise each inertial time series (ax, ay, az, gx, gy, gz) to have zero mean and unitary standard deviation. In the literature, this normalisation procedure is often referred as *z-normalisation*, and we will stick with this name.
- For each inertial channel of each recording, compute the moving standard deviation with a window length of 50 time steps.
- Compute the maximum value of each moving standard deviation time series ($maxMovSTD$).
- Compute the mean (M) and the standard deviation (S) of the resulting $maxMovSTD$.
- Label as outliers those recordings presenting $maxMovSTD$ values exceeding $M + 3S$. Assuming a Normal distribution of $maxMovSTD$, these samples labelled as outliers would appear with probability less than 0.3%.

Figure 6 shows the computed $maxMovSTD$ along with the orange horizontal line setting the $M+3S$ threshold. From the same picture we can see on the right plot that the Normal assumption is kind of justified for $maxMovSTD$ values. In Table 1, we can observe that the g_z and g_x time series are by far the most prone to be corrupted, and that accelerometer data tend to be less corrupted than gyroscopic data. Moreover, Scene 1 is by far the most corrupted. This could be explained since it was the first ever recording for each person. On the contrary, Scene 4 does not present corruptions, at least according to the

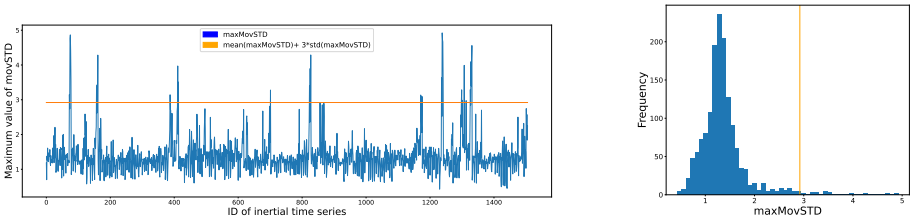

Figure 6: Results of *maxMovSTD*

Table 1: Number of corrupted time series according to *maxMovSTD* strategy.

Scene-Channel	a_x	a_y	a_z	g_x	g_y	g_z	Tot
Scene 1	2	1	3	5	3	4	18
Scene 2	0	1	0	1	0	2	4
Scene 3	1	0	1	1	1	2	6
Scene 4	0	0	0	0	0	0	0
Scene 5	0	1	0	1	0	3	5
Tot	3	3	4	8	4	11	33

maxMovSTD strategy; this might be trivially the position less sensitive to rib cage movements, and also body rotations being the closest to the human body's barycenter. Finally, note that the number of corrupted data of the others Scene 2, Scene 3 and Scene 5, are comparable with each other.

2.4 Pulsoximeter data

In order to check corruptions in the pulsoximeter data, we employed the following procedure.

- seek for zero length time series; 4 were found.
- then, seek for values of heart rate outside the range of 40–120 bpm; 5 were found.

Summarising, a total of 9 pulsoximeter recordings were corrupted. In Fig. 7 are plotted the five hr time series presenting nonsense values. In green the horizontal line of 120 bpm, while in orange the horizontal line of 40 bpm.

3 Anomaly detection

In this section we explored uncorrupted data to check whether we can find patterns or insights in the given data. For the purpose, we implemented two machine learning algorithms, one is a clustering-based (unsupervised) technique, and one is an autoencoder-based technique (supervised).

3.1 kMeans with Dynamic Time Warping distance

The unsupervised one is based on the kMeans clustering with Dynamic Time Warping (DTW). Basically, for the calculation of centroids in the kMeans algorithm, we replace the Euclidean distance with the DTW distance. The number

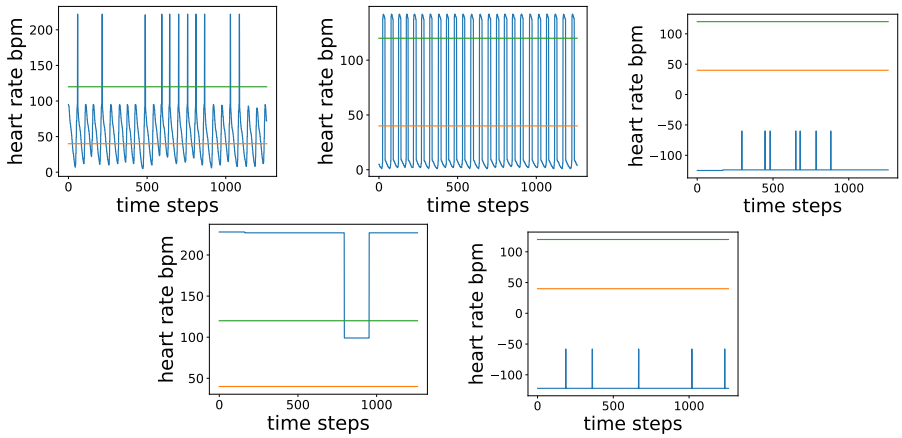

Figure 7: Nonsense recordings of heart rate signals.

of clusters is set to $k = 2$, ideally one should represents the "normal" recordings, and the other the "abnormal" ones. The procedure is summarised below.

- Extract from the inertial time series dataset (excluding the corrupted series individuated in section 2.3) the last 1090 time steps, this ensures to process all time series of the same length. We opted for 1090 in order to include all inertial time series, also those with long transients exceeding 340 time steps.

- Normalise each inertial time series (ax, ay, az, gx, gy, gz) to have zero mean. We did not rescale them to unitary standard deviation since the amplitude of the time series might be an important feature for the detection of anomalies and patterns. On the contrary, we would like to avoid the clustering algorithm to group time series together according to their mean values.

- For each scene and for each inertial channel, compute the 2-Means DTW clusters. We used the clustering.TimeSeriesKMeans function of the tslearn Python package.

- For each subject. If at least two different inertial channels of the same scene end into the less numerous cluster. Then, label those recording (i.e. corresponding subject's scene) as outliers.

This procedure gave disappointing results. We were not able to correlate the results with any label, e.g. male/female, smokers/nonsmokers, healthy/non-healthy, etc. Moreover, often the smaller cluster has size comparable with the larger. Furthermore, the algorithm gives quite inconsistent results when varying the initialisation seed. Also, normalising with a unit standard deviation did not bring any improvement. On the other hand, keeping the time series without rescaling their standard deviations makes the kMeans-DTW recognise sometimes few time series characterised by corruptions that were not detected by the method in section 2.3. We addressed this problem employing a supervised machine learning technique based on autoencoders in the next subsection.

This experiment suggests that this task may be too challenging to be solved for an unsupervised machine learning technique. However, other unsupervised techniques should be explored.

3.2 Autoencoder

After a first glance to the data, we realised that the *maxMovSTD* procedure did not manage to exclude all corrupted recordings. Therefore, after sifting inertial data with the *maxMovSTD* procedure, we trained an autoencoder with well-behaving recordings to detect further anomalous recordings. More precisely, we used the following procedure.

(I) Select all time series of a given scene and given channel, excluding the time series of the recordings with transients over 340 time steps, and perform z-normalisation on each channel.

(II) Among the selected time series of (I), sift out all those that exhibit a *maxMovSTD* higher than 1.5. The rationale of this would be to effectively extracting a subset of time series which are "well-behaving". This subset will represent the training dataset for the autoencoder.

(III) Split the selected time series of (II) in subseries of length 150 time steps. Lengths between 25 and 300 time steps have been tried, all giving similar results.

(IV) Train a autoencoder followed by a linear layer, to reproduce in output subseries of the training set defined in (III). After a few tries we found that vanilla sequential tanh layers of dimensions $(32, 2, 2, 32)$ were enough to reasonably solve the task. Optimiser was Adam, learning rate was 0.0001, loss was MSE. We explored training a Conv1D autoencoder obtaining similar results, so we opted for the vanilla one for sake of simplicity.

(V) Compute the maximum mean absolute error (MAE) obtained among the training subseries of (III), setting it as anomaly threshold. Hence, each time step of a time series which has a reconstructing MAE above this threshold is labelled as anomalous.

(VI) Evaluate the trained autoencoder on all the selected time series of point (I). Therefore, label as anomalous those time series presenting at least a number of 75 anomalous time steps, i.e. half the length of the subseries used for training the autoencoder.

Figure 8 shows some plotting of the inertial time series labelled as anomalous by the autoencoder, that were not discovered by the *maxMovSTD* strategy.

4 Distinguish the unhealthy from the healthy

In this section we aim to train a neural network to recognise healthy time series from unhealthy one, based on the only 4 subjects at our disposal labelled as unhealthy. We decided to build a balanced training dataset, despite the small number of recordings from unhealthy subjects in our hands. Therefore, for this task the training dataset will be formed by 3 healthy and 3 unhealthy inertial

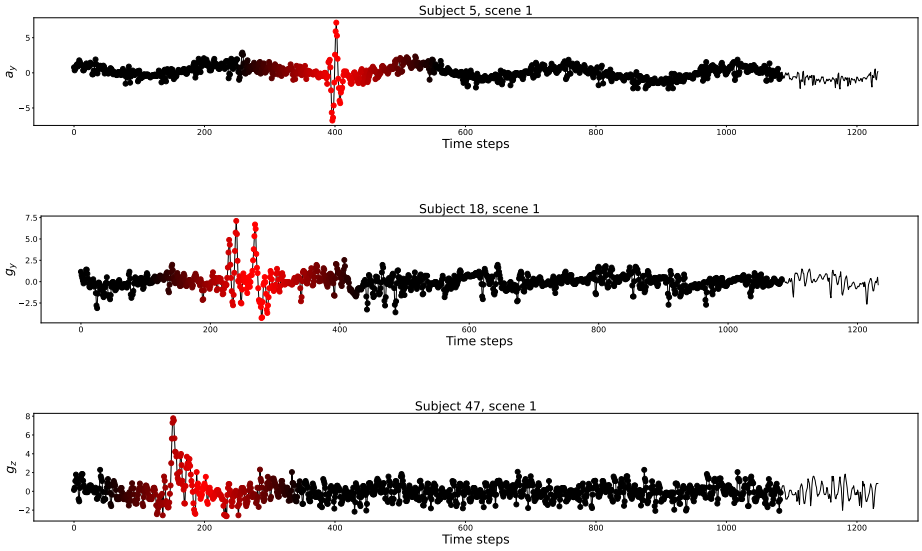

Figure 8: Mild corruptions in scene 1 labelled as anomalies by the autoencoder, the brighter the shade of red the stronger the likelihood of anomaly.

time series, one of which left out for validation. While the test dataset will be formed by the remaining 44 healthy time series, and the unhealthy one left out. The dataset is of course too small to expect satisfactory results, however — while waiting for additional data from the second phase of the project — we took the opportunity to build a processing pipeline for future use.

For the selection of the 3 healthy training subjects, we made sure that none of their 6 inertial time series were recognised as corrupted by the *maxMovSTD* strategy described in section 2.3.1 and by the autoencoder strategy of section 3.2. The 3 healthy subjects used for training of the models plotted in Fig. 9 were subjects identified by IDs 12, 36, 39. While the unhealthy used for training were subjects identified by IDs 28 (allergy), 30 (bronchitis), 38 (asthma), leaving out for testing the unhealthy subject with ID 37 (allergy).

We tried on every scene from 1 to 5, feeding the model with all the 6 inertial channels $a_x, a_y, a_z, g_x, g_y, g_z$. The scenes that present a higher success of reaching relatively low validation values are the first three, i.e. those corresponding to the chest positions. The curves of fig. 9 refer to scene 2.

We used LSTM, GRU, and SimpleRNN with Keras 2.4.0 and TensorFlow 2.3.0. In Fig. 9 are plotted the MAE curves on validation of the best models over 10 trials for all the three architectures.

Although the MAE curves of Fig. 9 depict a good scenario of learning, in reality the results on the test dataset were very poor, basically attributable to a random guessing for all the three models. In particular, the unhealthy subject left out for testing was half of the times predicted as healthy (incorrect) and half as unhealthy (correct). Similarly, all the other healthy subjects in the test dataset were labelled half the times as unhealthy (incorrect) and half the times as healthy (correct). Stacking more than one RNN layer did not bring any improvement in the test results.

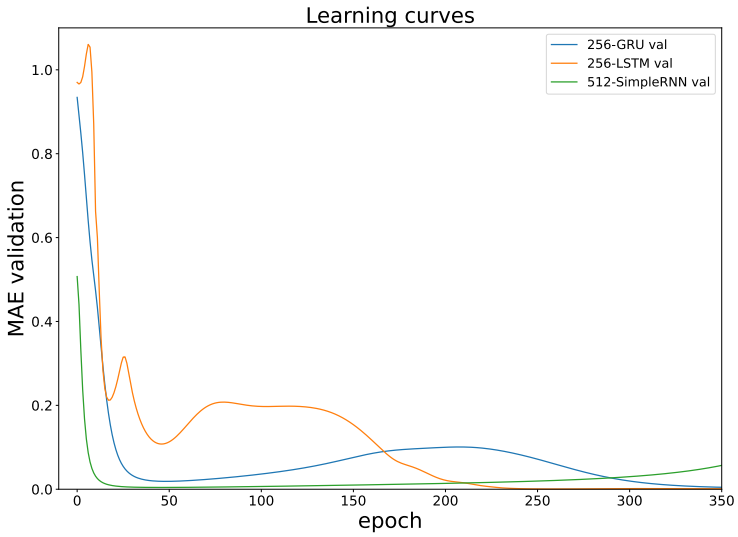

Figure 9: L2 regularisation with $\mu = 0.01$, batchsize 5, Adam optimizer with learning rate 0.001. In green SimpleRNN with 512 hidden units (266k parameters). In blue GRU and orange LSTM with 256 hidden units (respectively, 203k parameters, and 270k parameters). These results corresponds to recordings of Scene 2.

5 Train an RNN to generate HR in output driven with inertial data in input

In this section we explore to what extent we can extract useful information (like the heart rate) from inertial data without any preprocessing. Can we train an RNN driven in input by the inertial time series to generate in output a continuous signal that approximate the heart rate of the subject? Although estimating the heart rate is not the goal of this project, playing with this task helped us to individuate which recurrent neural network architectures are better suited for this kind of data.

For this task we focused on scene 3 (left chest) and channel a_z (outward-inward chest acceleration) since it is the recording combination that intuitively makes more sense for estimating the heart rate.

The procedure we implement is summarised as follows.

- Select all (scene 3, a_z) time series, extracting the last 1090 time steps, and performing z-normalisation.
- Among the 47 time series collected from healthy subjects, sift out all those that exhibit a $maxMovSTD$ higher than $M + 3S$ (as explained in section 2.3.1), and those corresponding to corrupted pulsoximeter data. Therefore, split in half for training and testing.
- train a Bidirectional LSTM of 64 hidden units followed by two tanh-Dense layers, of 10 and 1 neurons, to forecast the current heart rate given in input the past 150 time steps of a_z. We tried SimpleRNN, LSTM, GRU,

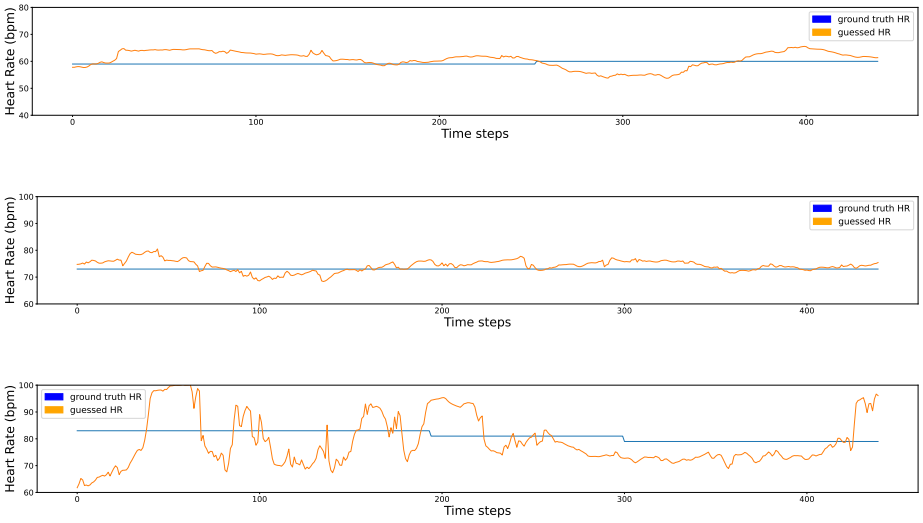

Figure 10: Examples of produced heart signals from a Bidirectional LSTM driven by the inertial time series a_z of the test dataset.

and Conv1D, but Bidirectional LSTM produced the best results. Adam optimiser with learning rate 0.001 and batch size 128 gave reasonably good results.

The target heart rate signal in bpm values has been rescaled in $(-1, 1)$ to facilitate the learning of the neural network. Precisely, the formula used is $\frac{x-80}{80}$, so that "typical" bpm values x between 40 and 120 are mapped in $(-0.5, 0.5)$. Then for visualisation purposes the generated output of the neural network has been transformed back to bpm values. In Fig. 10 few plots of the test session of the predicted hr (in orange) against the ground truth heart rate values (in blue) measured by the pulsoximeter.

In the first two cases the LSTM manages to produce reasonably good heart rate estimations, while in the third case the output oscillates wildly, but always around the ground truth. This experiments demonstrates that without any knowledge, or preprocessing of the a_z time series, even a basic neural network can learn to extrapolate meaningful values correlated with the heart rate.

6 Early conclusions

From the experiments run so far, we can see there are a few challenges to overcome. It seems hard to extract patterns from the inertial data and correlate them to given labels, and in particular to detect unhealthy people out of healthy ones. However, we are still in the early stages of the data acquisition. Hopefully, with more data and more fine tuning we can obtain better results. Moreover, we might consider the implementation of feature extraction algorithms to enhance the performance.

References

[1] L. Miglior. Implementazione di un'applicazione Android per il monitoraggio remoto di problemi cardio-respiratori. Master's thesis, University of Pisa, 2022.