Enosium

Track 2 : BED BASED BCG SIGNAL ANALYSIS

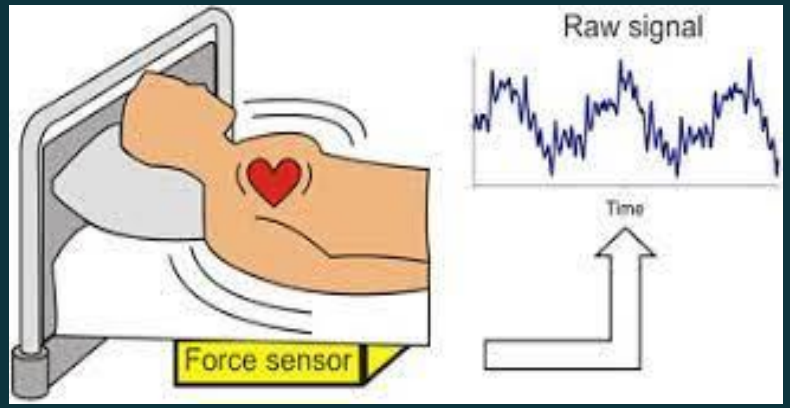
Robo_boys

- Tanvir Alam
- Sayan Acharya
- Ritodeep Sikdar
- Aditya Ganguly

Overview

What is BCG?

Ballistocardiography is a technique for producing a graphical representation of repetitive motions of the human body arising from the sudden ejection of blood into the great vessels with each heartbeat.



What was our approach for analysing BCG?

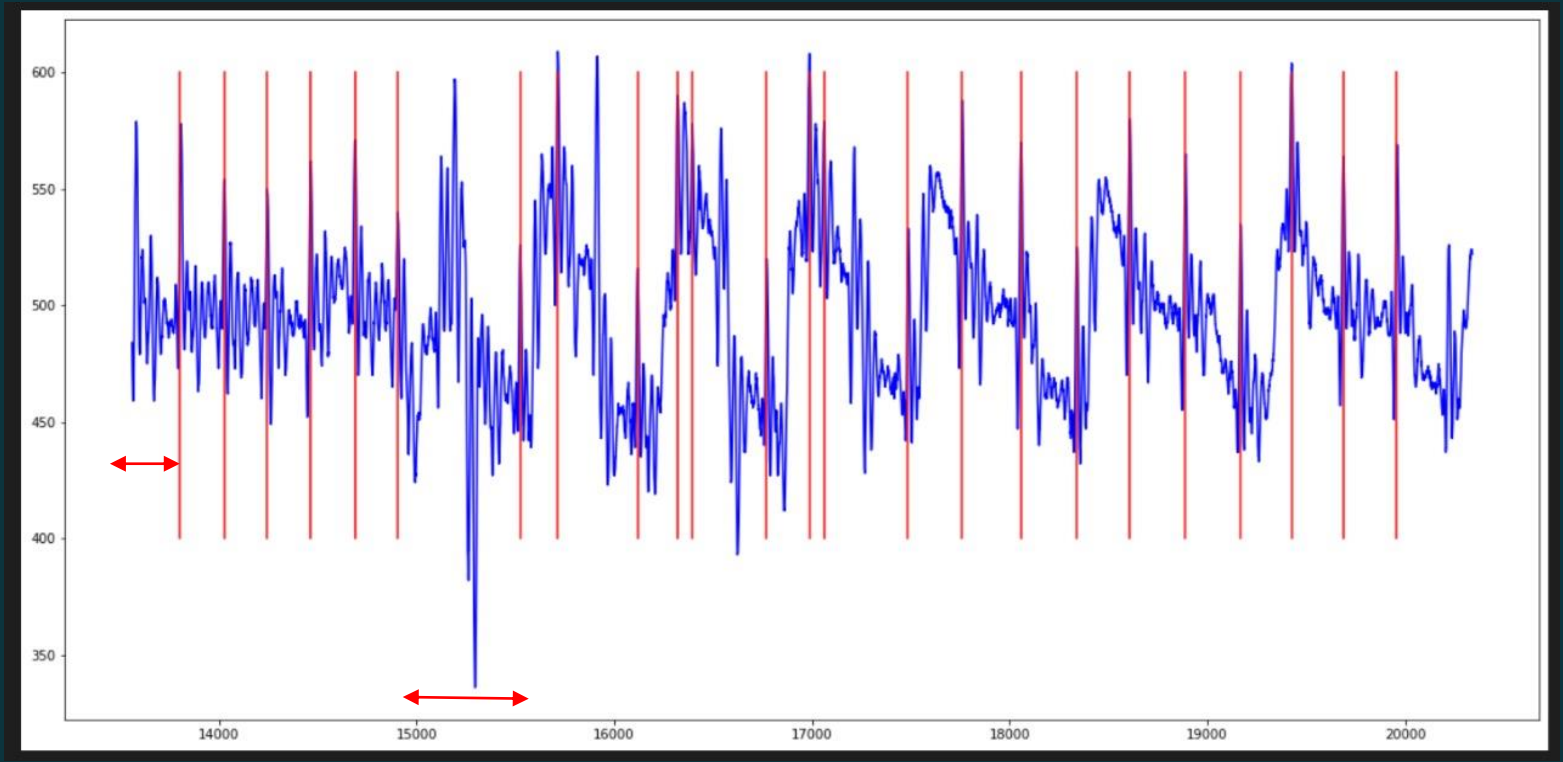
1. *We applied Sinusoidal Regression with Adaptive Learning for detecting J peaks and reducing noise in the signal.*
2. *Moreover we used time series signal analysis with Dragonfly optimization for detecting J peaks without considering noise.*
3. *We designed a Deep Learning model which tells us the presence of J peaks within a particular window length.*
4. *Finally, we developed algorithms for finding heart rate, breathing rate, motion artifact and bed occupancy.*

Data Preprocessing

The J-peak data was given as unix timestamps. We extracted the data and normalized into the unit system used in the bcg data csv. We found out that the frequency of the readings were 226 per second. So we normalized the jpeak data according to that scale.

J Peaks(Heartbeat) Identification

J peaks are those abrupt peaks in the continuous BCG data that tend to stand out from the data nearby them. But not all up-down portions are Jpeaks. It could be that someone has a general 'peaky' graph. So classification of J peaks is a tricky thing that is done by seeing if there is not a match with the surrounding data. For this purpose, we take a sliding window and identify all the maxima in that range. Now, we take those maxima and train a sinusoidal regressive function around that maxima and predict the possible height of that J-peak. If the predicted height is much smaller than the actual height then that means that point 'stands out' from the rest of them. This point is a probable J-peak. Now only this much checking is not enough so we used adaptive regression to classify a point as J-peak as there could be a lot of noise of many types (highly abrupt peaks, square waves, straight lines,etc.)



In this graph, we can see that in the middle and in the beginning there are some absurd peaks and inconsistencies. We were able to successfully remove the detection of those peaks so that our calculation does not get adulterated. (The vertical red lines through the graph signify possible J-peaks)

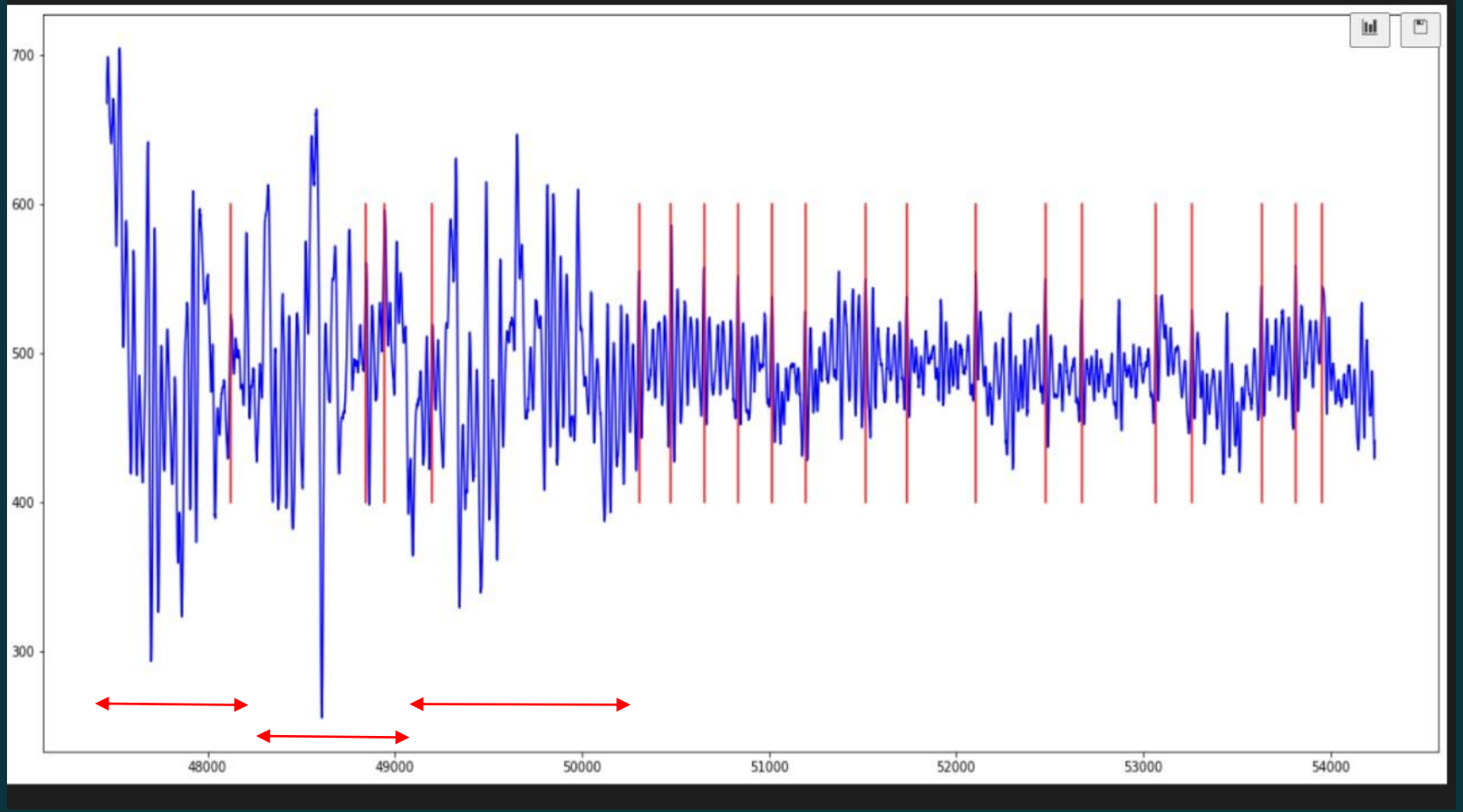
Regression and Noise Filtering

In the regression part, we used a sinusoidal function of the form:

$$f(x) = a0 + a1.\sin(x) + a2.\sin(2x) + a3.\sin(3x) + ...$$

Because this sort of function aptly matches the waveform produced by the heart. We trained the regression on a window of data surrounding a probable J-peak and then used it to predict the value. Now, to see if there is noise in that window, we calculated the std. deviation of all the points there and checked if the std deviation is too high or too low. Because too high std deviation is the result of abrupt high peaks or square waves and too low std deviation is the result of a non beating heart or an absent patient (empty bed)

We saw another interesting thing as well. The data was not following a constant trend throughout the whole range. At some places, the J-peaks were far away from each other or very closely clustered. So to increase the accuracy of our model, we introduced the idea of a type of Reinforcement Learning by adaptively changing the regression parameters like std_deviation_range or regression_window so that when there are a lot of spikes in a short period of time we increase our overall threshold and matching criteria to avoid taking all the surrounding maxima and similarly for other part, we relaxed the matching criteria so that J-peaks that are at a relatively smaller height can also be selected. So basically when our model matched a lot of data in a small range, we made the matching harder and when it did not match for a long time, we made it easier to match to better detect the J-Peaks of the given data. This increased the overall matching capability of our model and improved the overall J-peak classification.



The red horizontal lines show portions of noise and inconsistency trends in the data. The red vertical lines show J-peaks.

Adaptive Parameters

We kept a lot of parameters that control the adaption of the model to the incoming data. They are:

1. Reg_window_bound -> This parameter lets us clip the regression window if the window becomes too long or too small
2. Reg_div_const -> This parameter lets us calculate exactly at which rate we should increase the regression window depending on how long ago the last J-peak was
3. Thres_div_const -> This parameter controls how adaptively the thres_div_const should change with the reg_div_const
4. Jpeak_length_range -> This parameter shows us after getting how far from the last_match we should tune the adaptation
5. std_div_adjust_range/permit_range -> these parameters lets us handle the std deviation adaptively.

BCG Signal Processing using Global Maxima

We applied another interesting approach for finding the maxima and J-peaks thereafter using a peak finding algorithm combined with a state-of-the-art optimization algorithm.
We find all the peaks in the BCG signal and then identify J peaks by using a hyperparameter which calculates the difference in heights of consecutive peaks. Normally, a J peak will be the highest peak amongst its immediate neighbouring peaks. We made an objective function and used Dragonfly Optimization on this parameter to minimize the loss of J peaks. Now we adjusted the spacing of predicted J peaks accordingly.

Using this prediction of J peaks we were able to calculate the heart rate and breathing rate of the subject using the process mentioned earlier. The predicted and actual heart rates were found to be very similar with only 4-5% margin of error. This method was found to be working accurately with both subjects having significant motion artifacts and those having negligible motion.
This approach works faster as it ignores noise in the data whilst gives accurate predictions.

Deep Learning Model for Detecting J peaks

We use tensorflow and deep-learning to develop a model to predict whether a given range has a J-peak or not. We take as input a continuous range of length 50 from the data, and as output we check if that range has a J-peak within it. To do this we maintain the location of the previous J-peak for each point. If the starting and ending range has the same previous J-peak, the given range has not J-peak contained within it, otherwise there is a J-peak.

Along with this classification task , we add a regression task, to predict the relative location of the J-peak from the start of the range. We saw this provided better model performance and improved training stability.

Heart Rate

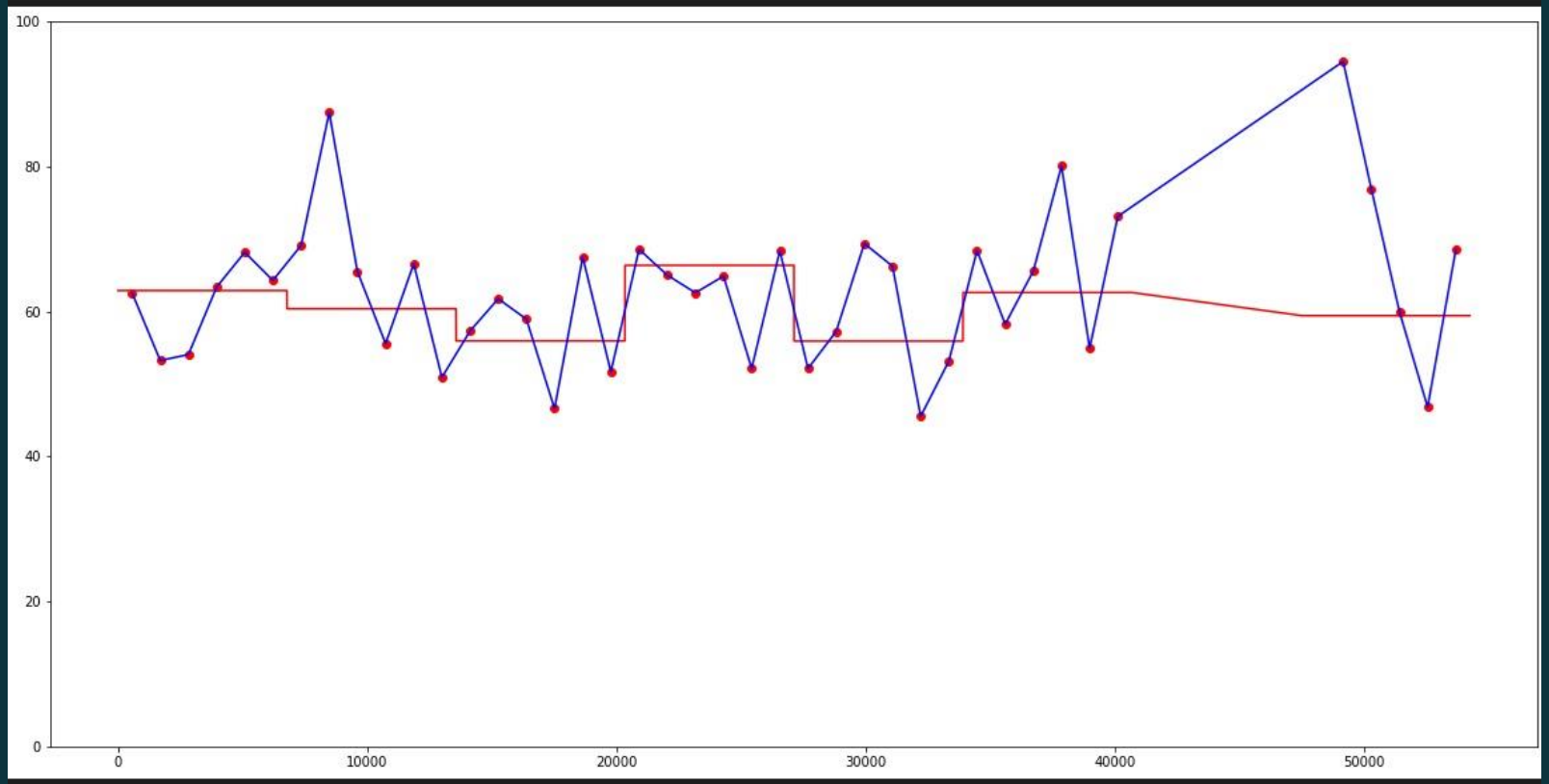
To calculate the heart rate, we take all the J-peaks and calculate the distance between them in seconds. Now the idea is that the distance between two J-Peaks is the time taken for a heartbeat.

So, we take that distance, say d.

Now 1/d shows exactly how many heartbeats occur in a second.

So, 60/d gives us the heart rate in per minute.

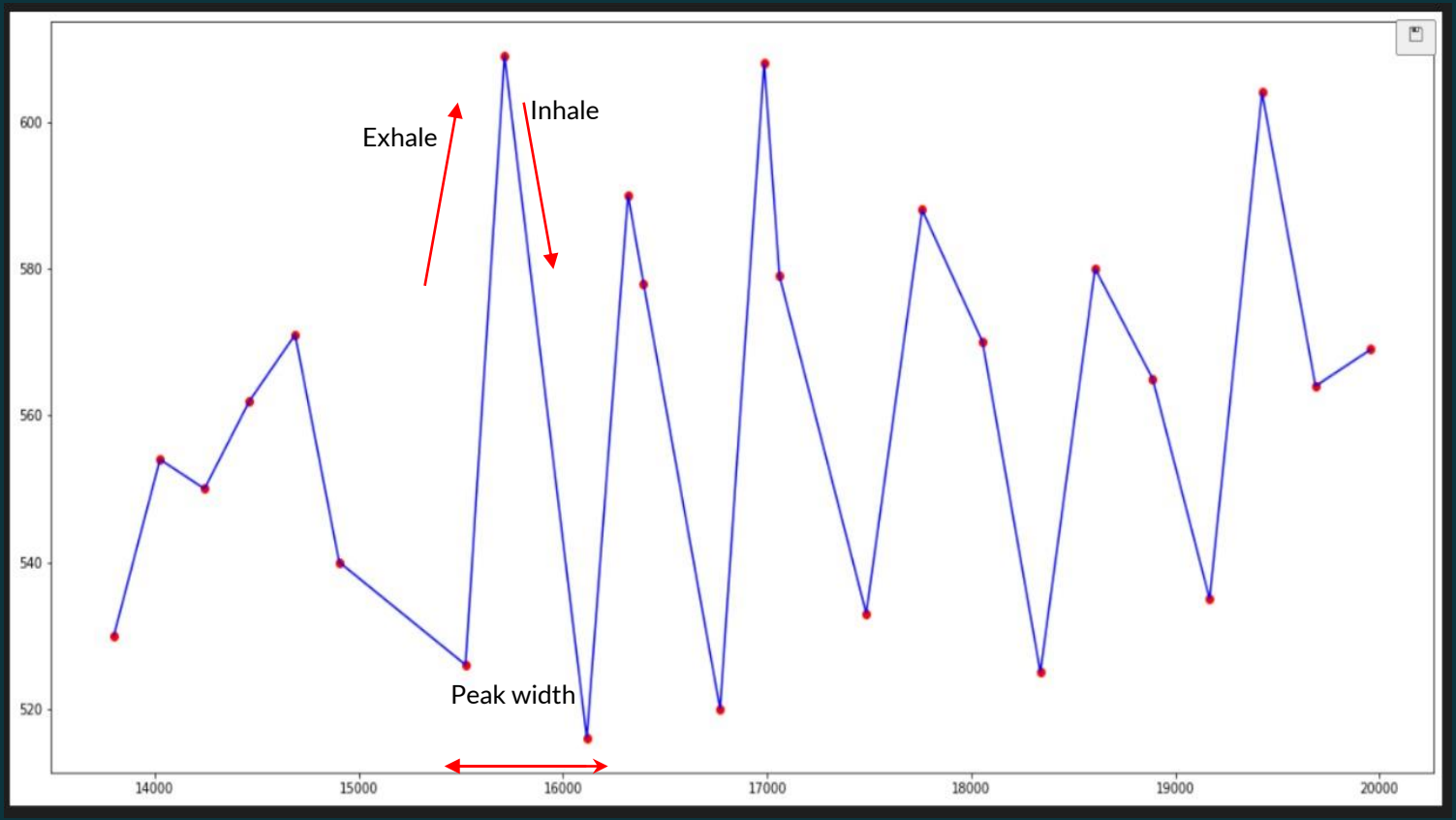
Now, if we see that if there are some d s that are too much away from the others, then we statistically remove them to get a more accurate value of the heart rate.



This is the a graph for heart rate. The blue lines show the heart rate within a 5 second range and the red line shows the more average heart rate over a range of 30 second.

Respiratory Rate

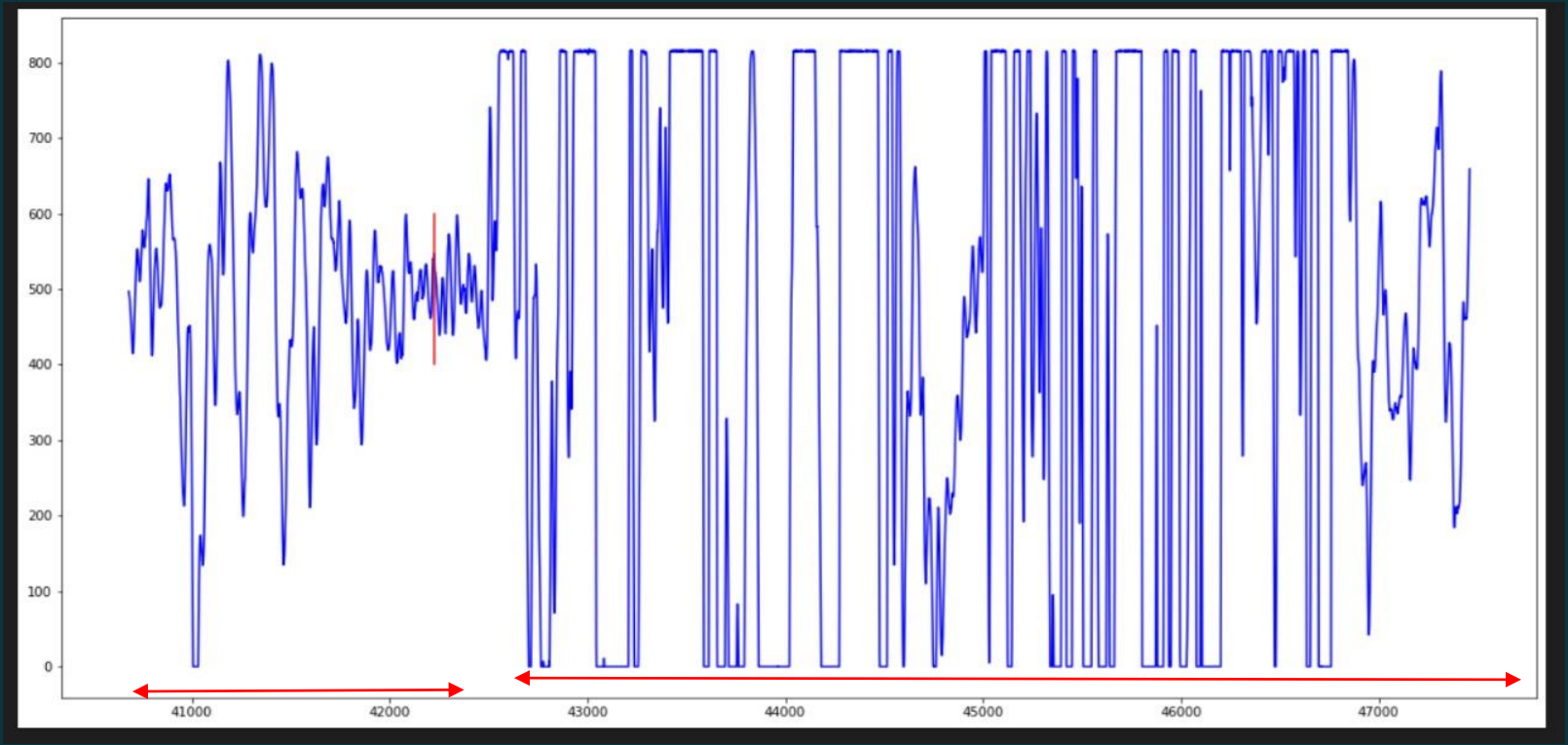
Calculating the respiratory rate was one of the biggest challenges of the whole task because there does not exist any clear, hard and fast rule to get the respiratory rate from the heart rate data or BCG data. Most methods give the idea of dividing the heart rate by 4 or 5 to get an average breathing rate but that method does not always work especially if someone is sleeping or exercising. So we did some research and found out that, during the exhaling period the bcg data rises and then during the inhaling period, it goes down. This cycle repeats over every breathing cycle. So to calculate the length of an average breathing cycle, we saw for how long the J-peak values were rising and that was our exhale-cycle. Similarly we calculated the inhale-cycle and added the length of the two cycles. That gave the length of a breathing cycle. Then we averaged those lengths over 30 seconds to get the respiratory rate.



This graph shows us the J-peaks (those red points are the J-peaks). We took those maxima s and minima s and calculated the peak-widths as distance from one minima to another minima. In this graph, the ascending part is the exhalation and the descending part is the inhalation. Addition of these two parts give the whole cycle length.

Motion Artifacts

Motion artifacts are detected by checking the portions of the data where we see huge changes in the pattern of the BCG data or presence or square waves. We classified the motion artifacts by noise detection discussed the previous slides. We classified motion artifacts depending on their deviation from the rest of data. If the std deviation was very high then it was of type 2 motion artifact and otherwise if the deviation was noticeably there but not was not of a very high value then we classified that as type 1.



These parts show the square waves that are of heavy motion induced noise. There are no detected J peaks in the noisy portions.

Bed Occupancy

Bed occupancy can be accurately predicted on the basis of whether there is a valid BCG pattern in the data which can be done in the way we discussed in the noise detection and motion artifacts detection portion. Now apart from that, if we saw that std deviation of a portion of the data was too small (close to zero) then that means that person's bcd data was flatlining or was not there to begin with. That gave us the bed occupancy.

Concluding Remarks

We are glad to have participated in this hackathon. We learned a lot of new things while analysing this Bed Based BCG Data and predicting logical results like heart rate, breathing rate, etc.

Our solution can be fine tuned further with maybe a different optimization algorithm or using a modified algorithm. The problem statement interested and intrigued us at the same time and we loved working on it.

We would like to further improve on our approach and use Machine Learning for solving medical problems.

For viewing and inspecting our solution codebase you can refer to the Github link given below and in description of the video.

https://github.com/FineTunersJU/enosium_track2