ECG Diabetes detection

Biomedical Signals Processing coursework

`by Max Bilyk`

Ukrainian Catholic University

APPS@CS

Lviv 2021

Introduction

Diabetes Mellitus (diabetes for short) is a clinical condition characterized by hyperglycemia which is widespread. Hyperglycemia is the condition in which there is not enough insulin in the body to deal with large amount of glucose present. It can not be cured, it only can (and should) be managed, otherwise it leads to great health problems.

- The number of people with diabetes rose from 108 million in 1980 to 422 million in 2014
- In 2019, diabetes was the ninth leading cause of death with an estimated 1.5 million deaths directly caused by diabetes.
- The timely diagnosis of diabetes is of great importance.

Now, when the usage of wearable devices, such as smartwatches or fitness trackers is widespread, it is easy to obtain biomedical data in non-invasive fashion. We should analyze ECG data because it is relatively easy to obtain. For example, we can measure ECG with Apple Watch or another device daily.

ECG and Diabetes

Often enough cardiovascular autonomic neuropathy (CAN) is caused by diabetes. It affect heart rate and blood pressure which leads to diminished Heart Rate Variability (HRV). In general, we can say that diabetes has significant negative effect on blood-vascular system. So, it is reasonable to use HRV (obtained from ECG) to detect diabetes, because HRV is indicative of blood-vascular system disorder either caused or exacerbated by diabetes.

Data Description

Dataset is collected in the context of the `D1NAMO` project, and consists of a set of **Electrocardiogram** (ECG -- signal of our interest), **Breathing**, **Accelerometers** signals, information about glucose levels and annotated food pictures. This project claims to aim at providing non-invasive diabetes management through analysis of data (signals) obtained by wearable devices.

Dataset was acquired on study conducted on 29 patients:

- 20 healthy
- 9 diabetes (Type 1).

Signals were taken in real-life conditions, with **Zephyr BioHarness 3** wearable chest-belt:

It measures ECG at a rate of 250Hz

Data is noisy and requires filtering

- Prerequisites (for participant to be enrolled):
    - >= 18 years old
    - Speak French
    - no significant psycho-social disability

Statistics of the dataset:

	Total	Average per participant
`Full dataset`		
Participants	29	
Signal recording	~1550h	~53h
`Subset of healthy participants`		
Participants	20	
Signal recording	~1100h	~55h
`Subset of participants with diabetes`		
Participants	9	
Signal recording	~450h	~50

- Participants are from different countries: **Switzerland**, **Italy**, **Spain**, **Russia**, **Australia**, **Mexico**, **Mauritius**, **Britain**.
- Age: 26-45

This dataset is really valuable because it provides real-world data, from non-medical grade devices which we are interested in.

Possible use cases (for `D1NAMO`)

- Explore relationship between different signals. For instance, **Breathing** with **ECG**.
- Develop algorithms suitable for usage in real-life conditions because data was taken not from medical grade but from wearable devices. Of course, this may require filtering and per-processing.
- Use in Deep Learning, Machine Learning application, because the amount of data is significant

Related Work

2. [Diabetes Detection using ECG signals: An Overview]

  - **Machine Learning**

    ML is widely used in Artificial Intelligence field, so there is no wonder that it is also commonly applied to signal analysis and processing. Moreover, ML algorithms helps us understand hidden patterns in data and made accurate prediction based on them. To work properly (efficiently) ML algorithms should be fed with right features. To select appropriate features a broad domain knowledge and deep understanding of signal under classification is required.

    There are many way in witch we can approach our detection problem. First of all we need to find appropriate domain to represent signal.

    - Frequency domain -- analyze all available frequency components (harmonics) present in the **HRV**
    - Time domain -- statistical calculation of mean and variance of RR interval of **HRV** data. Fourier transform, Wavelet transform.
    - Others

  - **Deep Learning**

    Deep learning is an improvisation of machine learning and it is suitable to high dimensional data and for complex **AI** problems. The shortcomings of machine learning (not intuitive, multidimensional features, domain knowledge required) led to development of deep learning. All the explicit feature-related processes found in the conventional machine learning networks are implicitly performed in deep learning networks. Deep networks self-learn from the data and its efficiency is much better compared to the traditional feature extraction networks.

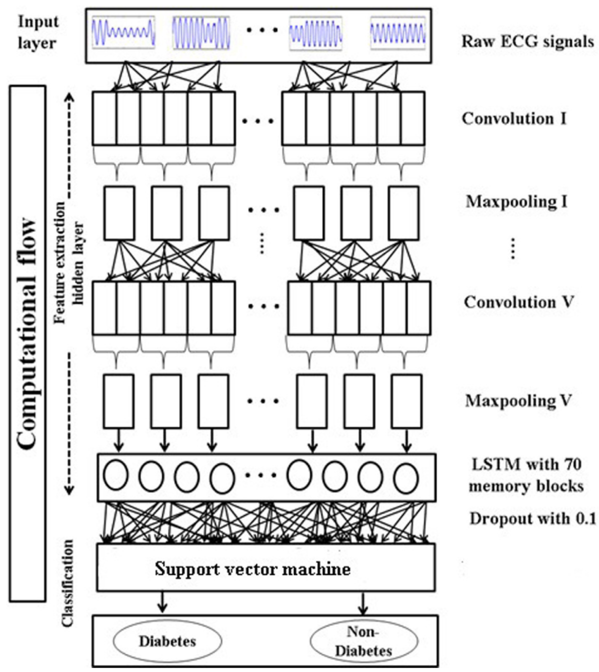
In this project, I will use **DL**, as in general it give better performance, and no deep domain knowledge is required for solving problem.

2. [Real-time QRS detector using Stationary Wavelet Transform for Automated ECG Analysis]()

   It does not matter whether to use **ML** or **DL** for analysis, we need to use some algorithm for extracting Heart Beats from ECG, then computing **HRV** obtained data. So, I decided to implement algorithm described in this paper for reaching this goal. Moreover, Wavelet Transform some parameter that we can play with, for example, ***wavelet mother function*** and ***number of decomposition steps***.

3. [Diabetes detection using deep learning algorithms]()

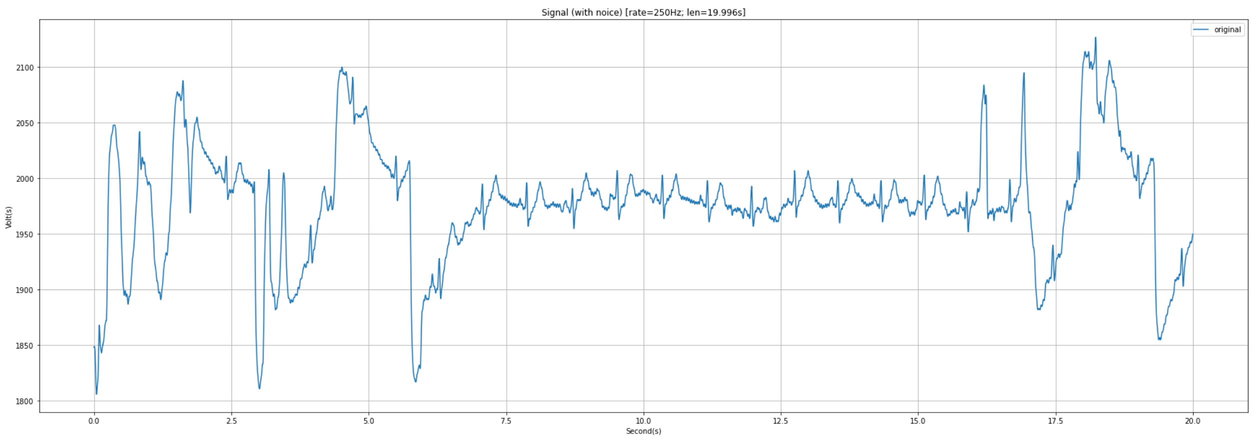
   In this paper describes Convolution Neural Network (CNN) I will implement.



Data manipulation

1. Heart Beat detection with implemented algorithm

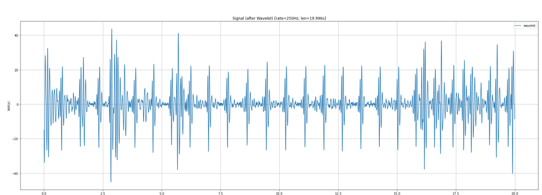
- `Original signal`



Step I — signal preprocessing

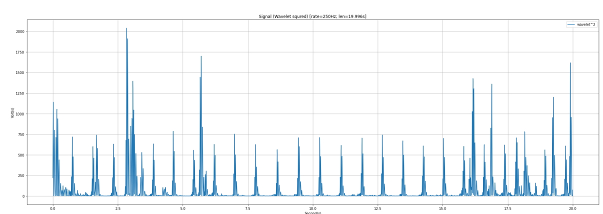
Resample it to 80Hz, do Wavelet transform and resample it back to the original sample rate — 250Hz. It is how it looks like:

- `Transformed signal`



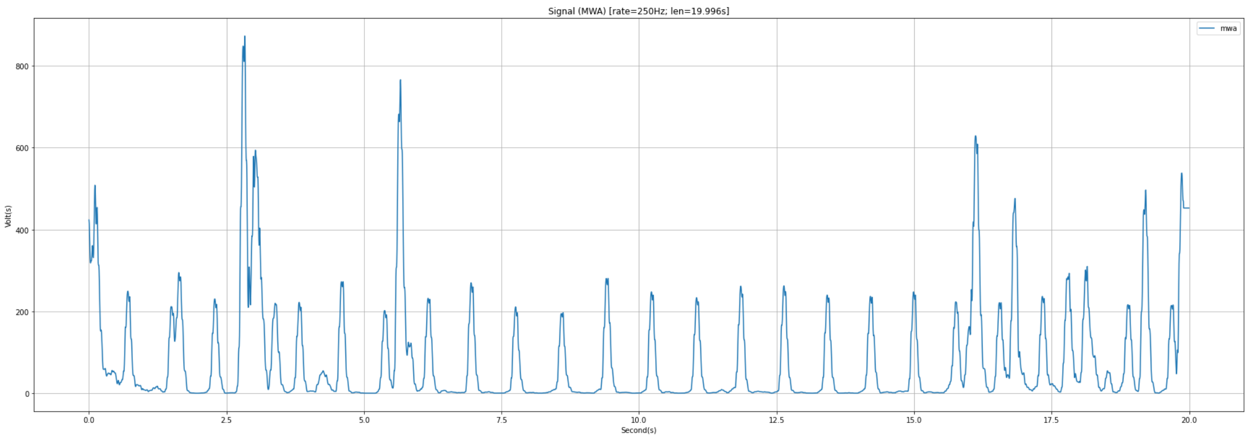
Then we square transformed signal:

- `Squared transform`



Apply averaging with window size/duration 0.1 second ~ 25 samples:

- `Smoothed (Moving Window Averaging)`



- Finally, normalize signal between 0 and 1

Step II — Heart Beat detecting

Algorithm (with some, maybe not even useful modification)

This algorithm works with processed signal:

For first 10 seconds

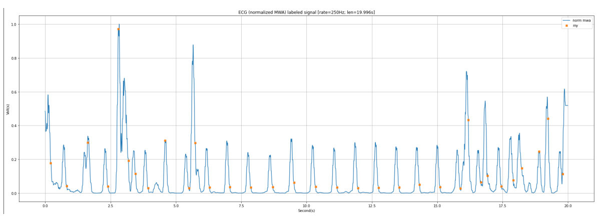
1. Calculate peaks, which amplitude exceeds certain threshold and located not exceeding certain density. Calculate their intervals
2. Find intervals that are longer than multiple of constant value and standard deviation of last 5 intervals. Then determine peaks that fulfill lighter condition.
3. Calculate new peaks and update intervals
4. Update threshold based on obtained data
5. Calculate final position of peaks (take the greatest peak in nearest area of current one). I used original signal to do this step, as it show better results with signals with lower noise
6. Recalculate some threshold (standard deviation of 6 last peaks)

Process next 3 second of signal and continue in the similar way

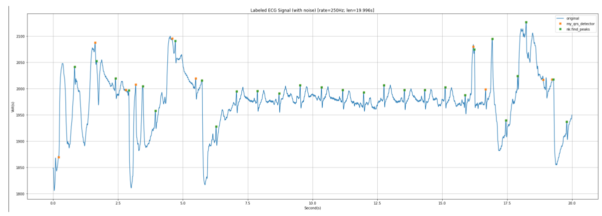
1. And so on...

It should be mentioned, that my implementation works really slow. Should rewrite it using `numpy` arrays exclusively, some day.

- `Labeled (Smoothed)` — the final decision on the location of peak (+-150 milliseconds) is based on the original signal (not transformed)



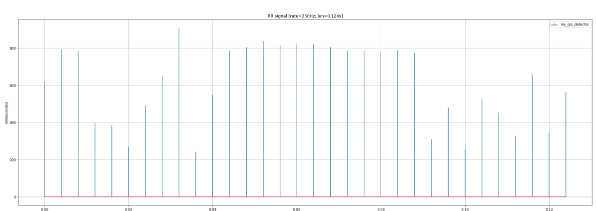
- Comparison of labeling with `neurokit2.ecg.ecg_findpeaks` library function find_peaks()



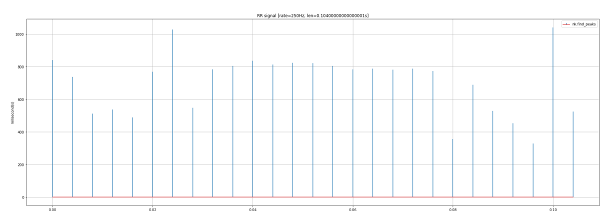
- Do observe that positions of labels are almost or just the same. From 5.5 to 16th second they are identical

---

- HRV (from custom implementation)



- HRV from `neurokit2.ecg.ecg_findpeaks`



- they look pretty the same

It is not that it works perfect with any amount of noise present, but I would say it works well enough.

Classification

Now, I shrink data into portions of 180 RR-intervals. Let's say I have `36151` rows of this interval set , I label it and normalize.

After that I divide data into training and testing portions:

- `.7` — training
- `.3` — testing

Labeled data of size [181 x 36151]

	0	1	2	3	4	5	6	7	8	9	...	171	172	173	174	175	176	177	178	179	lbl
0	584	548	556	524	548	468	504	500	504	512	...	528	564	468	492	508	524	452	524	500	0
3	648	776	772	756	108	652	728	728	620	756	...	632	108	636	108	644	108	620	108	620	0
5	872	108	908	108	872	124	880	1000	928	108	...	664	632	636	688	636	672	632	628	668	0
7	720	712	112	596	716	736	732	736	37084	112	...	112	680	112	680	796	792	764	768	824	0
8	760	732	700	696	736	852	932	888	780	784	...	960	37544	992	944	108	784	108	724	108	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
36140	164	376	108	420	184	332	120	256	252	208	...	444	236	212	184	304	108	268	252	632	1
36141	828	824	800	812	792	768	732	720	700	692	...	688	108	792	904	884	868	852	112	752	1
36143	112	600	708	712	704	704	704	700	700	704	...	692	712	700	692	696	688	688	688	676	1
36146	108	628	120	624	124	612	108	616	108	624	...	744	732	744	760	756	756	740	708	692	0
36150	776	108	692	108	732	828	820	792	724	120	...	724	736	700	696	108	664	108	772	108	0

Data is divided into supposedly equal to 3 minutes measurement portions on purpose. As it is project based on data collected from real-time measurement devices I think data should be processed correspondingly. 3 minutes, in my opinion is not that big time to disturb one waiting for their false/positive diabetes guess.

Structure of CNN (implemented on `keras`):

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_20 (Conv1D)           (None, 180, 64)           256
_________________________________________________________________
max_pooling1d_20 (MaxPooling (None, 90, 64)            0
_________________________________________________________________
conv1d_21 (Conv1D)           (None, 90, 128)           24704
_________________________________________________________________
max_pooling1d_21 (MaxPooling (None, 45, 128)           0
_________________________________________________________________
conv1d_22 (Conv1D)           (None, 45, 256)           98560
_________________________________________________________________
max_pooling1d_22 (MaxPooling (None, 15, 256)           0
_________________________________________________________________
conv1d_23 (Conv1D)           (None, 15, 512)           393728
_________________________________________________________________
max_pooling1d_23 (MaxPooling (None, 5, 512)            0
_________________________________________________________________
conv1d_24 (Conv1D)           (None, 5, 1024)           1573888
_________________________________________________________________
max_pooling1d_24 (MaxPooling (None, 1, 1024)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 70)                306600
_________________________________________________________________
dropout_4 (Dropout)          (None, 70)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 71
=================================================================
```

Training

I trained model for 10 epochs:

```
Epoch 1/10
791/791 [==================] - 101s 125ms/step - loss: 0.6384 - accuracy: 0.6917
Epoch 2/10
791/791 [==================] - 104s 131ms/step - loss: 0.5925 - accuracy: 0.7113
Epoch 3/10
791/791 [==================] - 117s 148ms/step - loss: 0.5957 - accuracy: 0.7111
Epoch 4/10
791/791 [==================] - 119s 151ms/step - loss: 0.5945 - accuracy: 0.7089
Epoch 5/10
791/791 [==================] - 111s 140ms/step - loss: 0.5822 - accuracy: 0.7151
Epoch 6/10
791/791 [==================] - 124s 156ms/step - loss: 0.5888 - accuracy: 0.7110
Epoch 7/10
791/791 [==================] - 130s 163ms/step - loss: 0.5957 - accuracy: 0.7075
Epoch 8/10
791/791 [==================] - 119s 151ms/step - loss: 0.5847 - accuracy: 0.7132
Epoch 9/10
791/791 [==================] - 120s 152ms/step - loss: 0.5864 - accuracy: 0.7122
Epoch 10/10
791/791 [==================] - 114s 144ms/step - loss: 0.5811 - accuracy: 0.7150
```

Evaluation

```
339/339 - 13s - loss: 0.5792 - accuracy: 0.7135
Accuracy:  0.713534951210022
Loss:  0.5791746973991394
```

I would say — awful results for binary classification. But what can we expect from dataset composed of measurement from 29 people with a lot of de-noising required (that actually was not approached in a right way, should probably use measurements from **accelerometer** for denoising)

Conclusions

Completing this project I was exposed to work with real-world data (not medical grade or cleaned). Now I fully understand that preprocessing and filtering is not just important for signal processing, it is essential and inalienable part for working with signals. I explored limitation of chosen dataset (noise, low variability of signal sources). Also I implemented algorithms based on paper mentioned above and tried to modify and tune it. Composing CNN almost as described in other paper was or was not successful, as it is hard to say based on results I obtained.

References

1. **Diabetes**
2. **Diabetes Detection using ECG signals: An Overview**
3. **Real-time QRS detector using Stationary Wavelet Transform for Automated ECG Analysis**
4. **Real-time QRS detector using Stationary Wavelet Transform for Automated ECG Analysis**