Comparison of acceleration python library on design and implementation of QRS detection module from ECG heart signal

C A Wicaksana¹, R Febriani² and F Muhammad¹
¹ Electrical Engineering, Faculty of Engineering, University of Sultan Ageng Tirtayasa
² Informatics Engineering, Faculty of Science Technology, University of Muhammadiyah Bandung

*E-mail: ¹cakraadipura@untirta.ac.id

Abstract. Electrocardiogram (ECG) signal is one of importance signal from our body that sourced from heart. There are many benefits that can be obtained from ecg signals, for example can determine whether sleepy/stress or not, and several diseases like arrhytmia, hypertension, heart failure, etc. In this research, we created sub-module for extraction feature for ecg signal using GPU Acceleration, but in this research only emphasis on QRS detection and peak detection from ecg signal. The input will be ecg Signals and the output will be array of peak in every row. There have been various research trying to extract or detect QRS and peak based on Pan-Tomkins Algorithm, but this research will make use python and will compare the acceleration using some library. The flow comprise five main step, (1) load ecg signal, (2) filtered ecg, (3) derivative from filtered ecg, (4) squaring from derivative ecg, (5) convolution squaring ecg, and (6) peak detection using Fiducial Mark. The overall module has been succesfully implemented and compared in python. The result show that computation using numpy is still better and faster for small array data. The Output of peak of array can be used to the next module.

1. Introduction
Topics of research in the area of heart signal is interest field. It is because heart signals contains many of information if it extracted well. For example, using HRV method, we can determine some of diseases or symptom from human body. For human heart diseases like valvular heart disease, bundle branch block, epilepsy, heart failure, apnea, hypertension, sudden cardiac death, myocardial infraction, myocardial ischemia, arrhytmia [1].

HRV or Heart Rate Variability is the physical state of a heartbeat condition in certain variables. The results obtained will be diverse and unique every second. Heart rate variability has several aspects such as time domain, geometry method, frequency domain and non-linear method [2]. Extracting HRV parameters, there are three steps to be done. First, ECG signal to RRI signal conversion; The second, Ectopic beats detection and correction; The Third, Ectopic beats detrending and resampling. In this research we only emphasis on acceleration using GPU of the first step, that is ECG signal to RRI signal conversion.

The substeps in the first step comprise six steps, they are (1) load ecg signal, (2) filtered ecg, (3) derivative from filtered ecg, (4) squaring from derivative ecg, (5) convolution squaring ecg, and (6) peak detection using Fiducial Mark. The comparison of acceleration will be in both squaring from derivative ecg and convolution squaring ecg because many literature mention that square and convolve will be
high-computing. This research will be based on previous research from [3-5]. It will be implemented for further research in advance using technique and some of platform software from [3]. All data used in this research are come from physionet.org (https://physionet.org/pn3/ecgiddb/) and the data was created and contributed by Tatiana Lugovaya [6].

2. Related Work and Previous Work
This research is programmatically based on [3, 5] and algorithmically based on [4]. As mentioned on [3, 5], all the computing is running very well without any problems, but we need to make it more faster because we think further about data for real time ECG measurement, which of course will have plenty of big data. In journal of [3], only gathering ECG data from ecg device sensor AD8232 trough cloud MQTT protocol. The ecg data will be saved on database and display on website, so that the patient and doctor can see the ecg data graphics. At the same time in [5] both Michał Sznajder and Marta Łukowska developed ECG QRS detector based on python by using ecg data from .csv file extension. They source code are available on github in link https://github.com/c-labpl/qrs_detector. The data have range value from 1.0 until 3.4 for the maximum ecg value for one channel. Data from physionet contains 90 person, in every person there are two times record data. Every record data contains three files extensions, they are .atr, .dat, and .hea.

3. Method
Algorithmically this research is based on [4], that is Pan-Tomkins Algorithm. The block diagram of Pan-Tomkins Algorithm, more or less like the following as described on [7] and [8]:

\[
\begin{align*}
    x[n] & \rightarrow \text{LPF} \rightarrow \text{HPF} \rightarrow \frac{d[\cdot]}{dt} \rightarrow [\cdot]^2 \rightarrow \frac{1}{N} \sum_{t=1}^{N} y[n]
\end{align*}
\]

**Figure 1.** The block diagram of Pan-Tomkins Algorithm[7].

3.1. LPF (Low Transfer Function)
This is the Transfer function of low-pass filter:

\[
H(z) = \frac{(1 - z^{-6})^2}{(1 - z^{-1})^2}
\]

3.2. LPF (High Transfer Function)
The is the Transfer function of high-pass filter:

\[
H_{hp}(z) = \frac{P(z)}{X(z)} = z^{-16} - \frac{H_{hp}(z)}{32}
\]

3.3. Derivative function
The transfer function of the derivative function as follow:

\[
H(z) = 0.1 \left( 2 + z^{-1} - z^{-3} - 2z^{-4} \right)
\]

3.4. Quadratic function
Quadratic function has function to make it become positive values. It will emphasize the high frequency signal, which is mainly the QRS complex. The function are here:
\[ y(nT) = [x(nT)]^2 \]  

(4)

3.5. Moving-window integration
Moving-window integration obtains wave-form feature information and the slope of R wave. The function are here:

\[ y(n) = \frac{1}{N} \sum_{i=1}^{N} [x(n-(N-1)) + x(n-(N-2)) + \cdots + x(n)] \]  

(5)

N is the width of the integration windows and depend on the number of samples.

3.6. Peak detection
For detecting peak on integrated measurement, in this research uses the Fiducial Mark and the source code from[9], but for further explanation about Fiducial mark detection of eeg qrs complex based on polygonal approximation. Further more explanation about Fiducial mark are in [10].

4. Proposed Function to be Compared
In this research, we will inject some library to few current function of computing that will lead to less time consumption or maybe high time consumption. The injection will be put on equation number (4) and (5). We will use python library to do computing process, they are both Numpy, Numba, and Pycuda[11].

4.1. Environment of development
The environment of software development using Windows 10 Professional, Anaconda version 4.7.5, Python 3.7.1, Numpy 1.16.4, Numba 0.44.1, Wfdb 2.2.1, PyCuda 2019.1, Cudatoolkit 10.1.168, CUDA 7.5. Software of text editor or IDE using PyCharm Professional Edition 2019.1 with valid license. On the other side for hardware development, we use Intel(R) Core(TM) i7-4510U CPU @ 2.00 GHz 2.60 GHz, RAM DDR3 12 GB, NVIDIA GeForce 840M 2GB Dedicated VRAM, Hardisk SSD 128GB.

4.2. Function and kernel
We inject one function using Numba and one kernel using Pycuda. The Numba function will be added as follow to equation number (4):

```python
@vectorize(['float32(float32, float32)'], target='cuda')
def VectorSquaring(a, b):
    return a * b
```

In the library setting, add line as follow:

```python
from numba import vectorize, cuda
```

Symbol “a” and “b” is the result vector from derivative vector using equation number (3). In python function, VectorSquaring, we just need to multiply it by itself.

On the other side one kernel .cu added to equation number (5) and called by using PyCuda as follow:

```c
__global__ void convolution(float *N, float *M, float *P, int Mask_width, int width){
    int i=blockIdx.x*blockDim.x+threadIdx.x;
    float Pvalue=0.0;
    int N_start_point=i-(Mask_width/2);
    for(int j=0; j<Mask_width; j++)
    {  
        if(((N_start_point+j)>=0) && ((N_start_point+j)<width))
        {
```
That kernel based on basic computing of convolution one dimension. For Example: there is an array one-dimension like this [0, 3, 4, 5] and one-dimension array filter like this [1, 2]. The computation will as follow:

\[
[0 \ 3] \ast [1 \ 2] = \begin{bmatrix} 0 & 6 \\ 3 & 8 \\ 4 & 10 \end{bmatrix} \Rightarrow \text{Sum all the array elements}
\]

Convolution one-dimension output = [6, 11, 14]

### 4.3. ECG raw data

All ecg data in this research are from physionet.org ([https://physionet.org/pn3/ecgiddb/](https://physionet.org/pn3/ecgiddb/)), but for this experiment will only 5 person of ecg raw data from one channel. To read data from physionet.org, we use python library called wfdb with version 2.2.1. The code added as follow to current source code:

```python
ecgrecord, field = wfdb.rdsamp(self.ecg_data_path, sampfrom=0, sampto=self.signal_length)
```

that function will return two variable, that is ecgrecord and field. Ecgrecord contain data of measurement, ecgrecord[0] is ecg data without filtered and ecgrecord[1] is ecg data filtered. The last variable field contain additional information about the person and ecg data.

### 5. Result and Discussion

The result of python program will sequentially presented as follow with signal frequency set to 300 and integration window set to 4:

| Person | Array Length | ecgrecord[0] (second) | ecgrecord[1] (second) | Δt |
|--------|--------------|-----------------------|-----------------------|----|
| 1      | 10000        | 0.001183              | 0.22673               | 0.21920 | 0.225547 | 0.218333 |
| 2      | 10000        | 0.001134              | 0.21073               | 0.22161 | 0.209596 | 0.220767 |
| 3      | 10000        | 0.001605              | 0.22401               | 0.22225 | 0.222405 | 0.221431 |
| 4      | 10000        | 0.001159              | 0.22550               | 0.21187 | 0.224341 | 0.210841 |
| 5      | 10000        | 0.001085              | 0.21919               | 0.22032 | 0.218105 | 0.219334 |

**Table 2.** Result of comparison in equation number (5).
As shown in Table 1, Table 2, and Table 3 the result shows for an average array length 10000 the library numpy is still faster than PyCuda and Numba. For small array numpy is still faster than GPU libraries. Numpy is still based on CPU not on GPU. Numba and PyCuda are acceleration libraries for large data of arrays. Actually, there are many tools of acceleration like CUBLAS, CUFFT, scikit-cuda that can be compared in next research.

This source code is available on author's GitHub.

6. Conclusion

Implementation QRS detection and peak detection were successfully implemented with source ECG data from physionet.org. The result shows that numpy is still faster than others. Though numpy is running on CPU platform, Numba and PyCuda are running slower than numpy because the data is still small and not large. In the equation number (4) we can see that the average of $\Delta t$ from signal I is $0.2232715$ and
\( \Delta t \) from signal II is 0.22076227. Both average signal I and signal II on the equation number (4), numpy is still faster. Lastly in the equation number (5) we can see that the \( \Delta t \) from signal I is 0.0050251 and \( \Delta t \) from signal II is 0.00522328. Both average signal I and signal II on the equation number (5), numpy is still faster.

7. Future Work and Research
All the output of this module will be the input for the next module on next research. The output of this module will be some peak from ecg signal. That peak indicate RR signal. This application can be implemented as a back-end service to compute and analyze ecg signal either offline or realtime signal from device. This module can be joined and implemented on the project research from [3]. If there is any inquiry or any advice, don’t hesitate to contact the member of researcher in this project.

References
[1] Timothy V 2017 DESIGN AND IMPLEMENTATION OF AUTOMATED DIAGNOSIS SYSTEM BASED ON HRV ANALYSIS AND MACHINE LEARNING (Institut Teknologi Bandung)
[2] Wijaya A I 2016 SHESOP HEALTHCARE: ANDROID APPLICATION TO IDENTIFY STRESS AND INFLUENZA USING TIME DOMAIN HRV AND SUPPORT VECTOR MACHINE KERNEL (Institut Teknologi Bandung)
[3] Rinanda Febriani, Sutadi Triputra, Ririn Suharsih C A 2019 Sistem Layanan Kesehatan Terpadu Home Care Digital Medic dan Wearable Device Monitoring System J. Sci. Technol. Enterpreneursh. 1 106–14
[4] Pan J and Tompkins W J 2007 A Real-Time QRS Detection Algorithm IEEE Trans. Biomed. Eng. BME-32 230–6
[5] Sznajder M and Łukowska M 2017 Python Online and Offline ECG QRS Detector based on the Pan-Tomkins algorithm 1
[6] Ary L. Goldberger, MD; Luis A.N. Amaral, PhD; Leon Glass, PhD; Jeffrey M. Hausdorff, PhD; Plamen Ch. Ivanov, PhD; Roger G. Mark, MD, PhD; Joseph E. Mietus, BS; George B. Moody, BS; Chung-Kang Peng, PhD; H. Eugene Stanley P 2000 Components of a New Research Resource for Complex Physiologic Signals PhysioBank, PhysioToolkit, and PhysioNet 101 215–20
[7] Aquino-Santos R, Martinez-Castro D, Edwards-Block A and Murillo-Piedrahita A F 2013 Wireless Sensor Networks for Ambient Assisted Living Sensors (open access J. Sci. Technol. sensors) 13 16384–405
[8] Nguyen T, Vuong N, Tran L D and Huynh T Van 2017 Detect QRS complex in ECG IEEE Conference on Industrial Electronics and Applications (ICIEA) (Siem Reap, Cambodia: IEEE) p 6
[9] Janko Slavic 2015 Peak detection algorithm and implementation 1
[10] Lee S, Jeong Y, Park D, Yun B-J and Park K H 2018 Efficient Fiducial Point Detection of ECG QRS Complex Based on Polygonal Approximation Sensors (open access J. Sci. Technol. sensors) 18 16
[11] Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P and Fasihe A 2009 PyCUDA: GPU Run-Time Code Generation for High-Performance Computing CoRR 19