Biomedical Signal Processing Final Report

School of Biological Science and Medical Engineering

Zhao Xiangyu

17374468

I. Introduction

Afib is known as the most common sustained cardiac arrhythmia, which affects almost 1-2% of the general population, and is associated with significant mortality and morbidity through association of risk of death, stroke, hospitalization, heart failure and coronary artery disease, etc. 2017 Physionet Challenge Dataset is a single-channel ECG signal dataset which includes 4 types of signals: normal, Afib, other rhymes and too noisy to recognize.

In this report, we will explore the internal patterns of normal, Afib and noisy signals from both time domain and frequency domain. Discrete Fourier transform (DFT) will be performed on chosen signals to observe frequency features directly, and power spectrum will also be plotted to observe the energy distribution of signals. Then, we design a FIR linear-phase digital filter to denoise original signals and observe its effect.

After above necessary preprocessing, we will utilize Hamilton R-peak segmentation algorithm to detect R-peaks of the signals, which will be used for calculating heart rate and extract related features. In addition, we will extract useful features from filtered signals to perform binary classification upon normal and Afib signals.

II. Aims

1. Plot normal, Afib and noisy signals in time domain.
2. Perform DFT on typical signals and yield power spectrum; observe signals from both time domain and frequency domain to find internal patterns.
3. Design a filter to denoise signals and observe the effect.
4. Calculate heart rate of both normal and Afib participants.
5. Design machine learning algorithms to classify normal and Afib signals.

III. Materials

2017 Physionet Challenge Dataset consisted of 8,528 short single lead ECG segments donated by AliveCor, which aims to encourage the development of algorithms to classify, from a single short ECG lead recording (between 30 s and 60 s in length), whether the recording shows normal sinus rhythm,

atrial fibrillation (AF), an alternative rhythm, or is too noisy to be classified. All the ECG recordings were sampled as 300 Hz and have been band pass filtered by the AliveCor device. Four classes of data were considered: normal rhythm, AF rhythm, other rhythm and noisy recordings, and all the data is labeled by a same group of experts.

Type	# recording	Time length (s)				
		Mean	SD	Max	Median	Min
Normal	5154	31.9	10.0	61.0	30	9.0
AF	771	31.6	12.5	60	30	10.0
Other rhythm	2557	34.1	11.8	60.9	30	9.1
Noisy	46	27.1	9.0	60	30	10.2
Total	**8528**	**32.5**	**10.9**	**61.0**	**30**	**9.0**

Table 1: Statistics of ECG data

IV. Methods

We choose some random signals from the original dataset to observe and process, rather than process the whole dataset. The standard of human selection is the quality of signal waveform in time domain, such as duration and signal-to-noise ratio. In this case, it will be easier to observe and find internal patterns of both normal and Afib signals with naked eyes, and will not be highly affected by unwanted noise. For every class we choose 3 typical signals to observe their internal patterns.

Supporting materials include figures which are not showed in the paper, and source code for the whole process. Time-domain plot, DFT and filtering are performed on Matlab R2018a. Heart rate calculation and classification are performed on Python 3.7 Anaconda release.

The whole procedure of the process is described in the flow chart below:

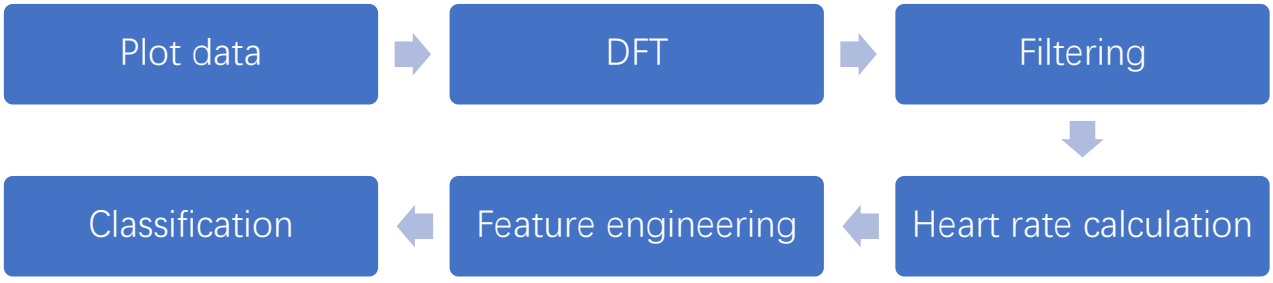


Table 2: Flow chart of the process

4.1 Plot in time domain

First, we plot raw ECG signals in time domain and observe its patterns.

Normal signals from left to right is: A00001, A00006, A00007; same below.

Afib signals from left to right is: A00004, A00009, A00027; same below.

Noise signals from left to right is: A00034, A00106, A00307; same below.

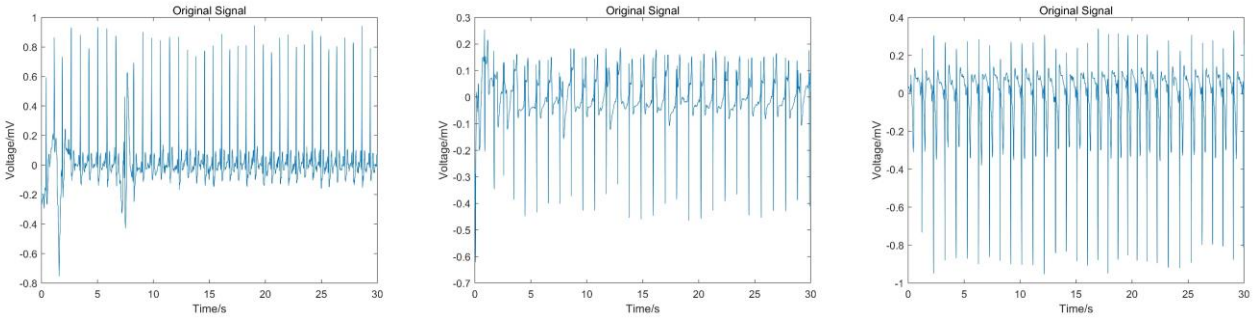


Fig 1. Normal signal in time domain

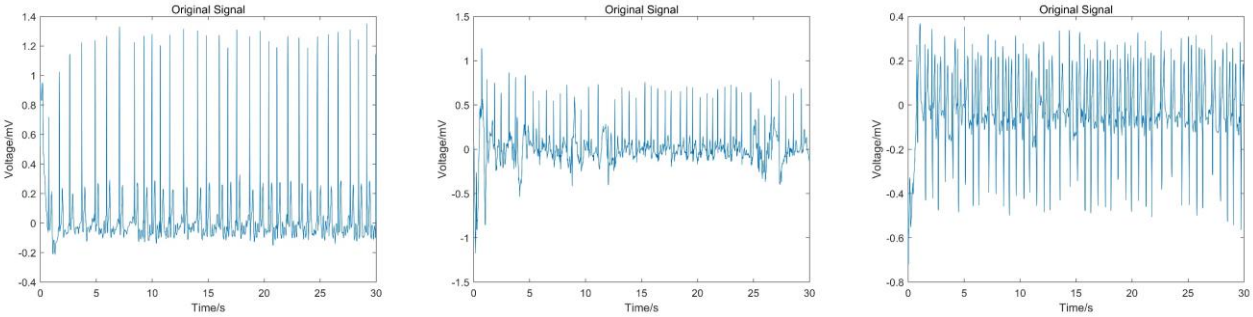


Fig 2. Afib signal in time domain

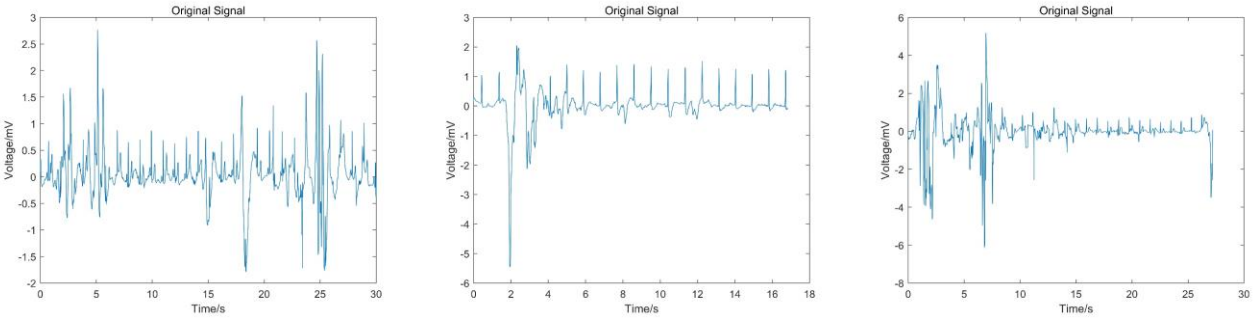


Fig 3. Noisy signal in time domain

From time-domain waveforms, patterns below can be observed:

1. Normal and Afib signals contain much less noise compared with noisy ones, but filtering may be still of benefit for further purposes.

2. In normal signals, R-peak intervals are well-distributed in time, which indicates a comparatively steady heart rate. While in Afib ones, R-peak intervals are random and appear uneven in time, which indicates cardiac arrhythmia's presence.

3. Noisy signals contain much noise which overwhelms internal patterns of original waveforms.

Filtering may be helpful to reduce noise interference and explore internal patterns.

4.2 Discrete Fourier transform

We perform DFT upon original signals by invoking function *fft* in Matlab, which returns a complex number array, which represents the frequency spectrum of original signal. In order to plot it, we will invoke function *abs* to acquire the module (absolute value) of frequency spectrum and plot the amplitude spectrum.

$$DFT: \quad X(k) = \sum_{n=0}^{N-1} x(n) e^{-jn\frac{2\pi}{N}k} = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

It should be notice that the DFT of a N-point sequence is also a N-point sequence. In order to acquire the frequency spectrum, we need to convert k into analog frequency f. The relationship is described below:

$$\omega = \Omega T_s = 2\pi \frac{f}{f_s}$$

$$i.e.\ 2\pi \frac{k}{N} = 2\pi \frac{f}{f_s}$$

$$i.e.\ f = \frac{k f_s}{N}$$

According to analysis above, we plot the amplitude spectrum below:

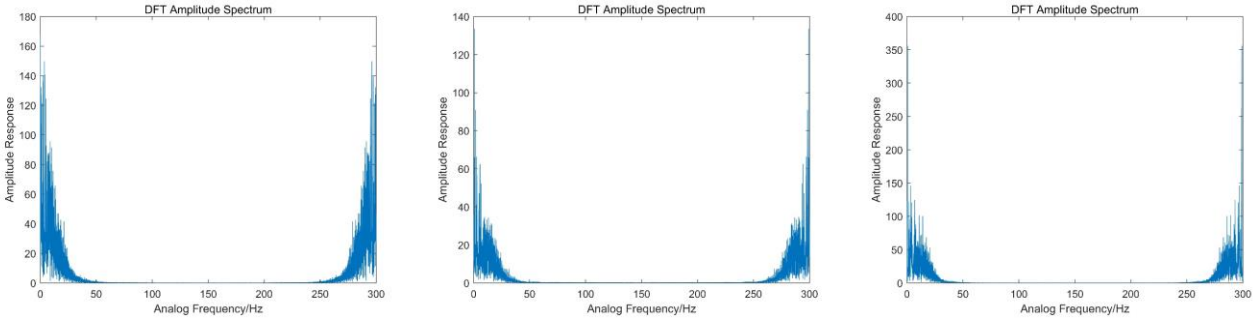


Fig 4. Normal signal DFT

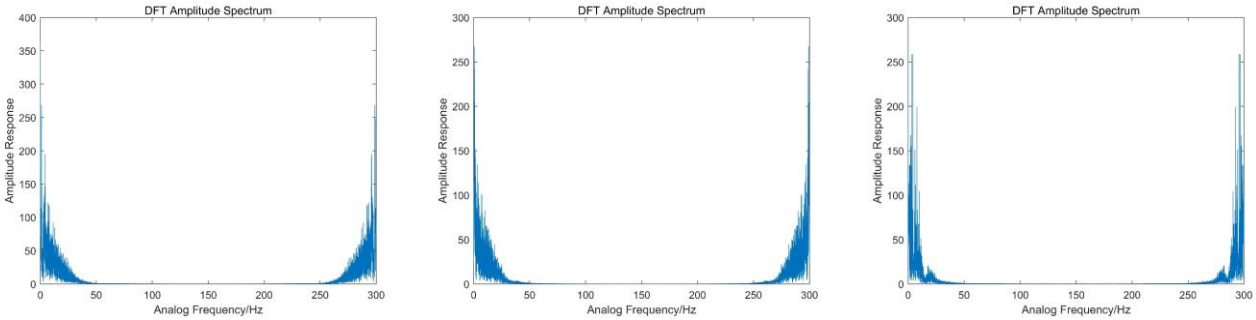


Fig 5. Afib signal DFT

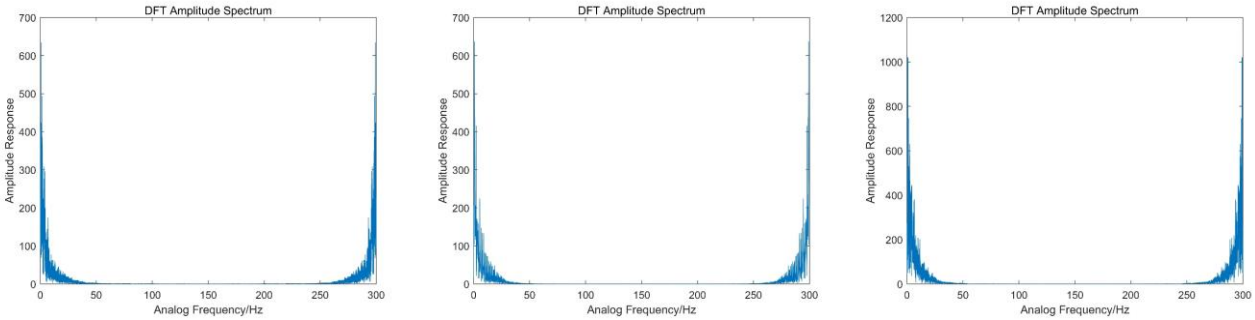


Fig 6. Noisy signal DFT

From the implicit periodicity of DFT, we invoke function *fftshift* to induce a zero-center frequency spectrum to observe the signal more clearly.

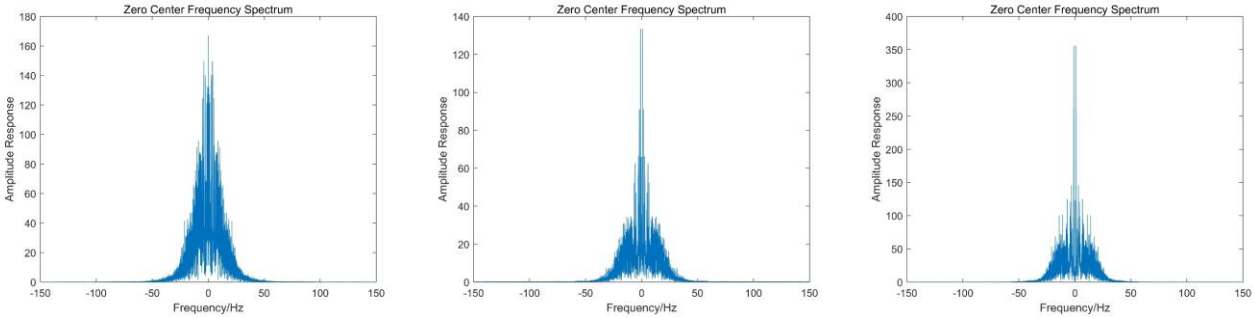


Fig 7. Normal signal zero-center DFT

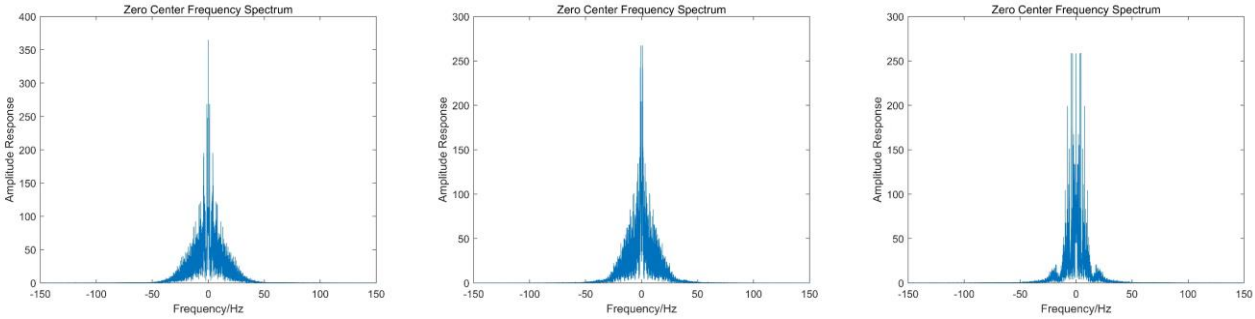


Fig 8. Afib signal zero-center DFT

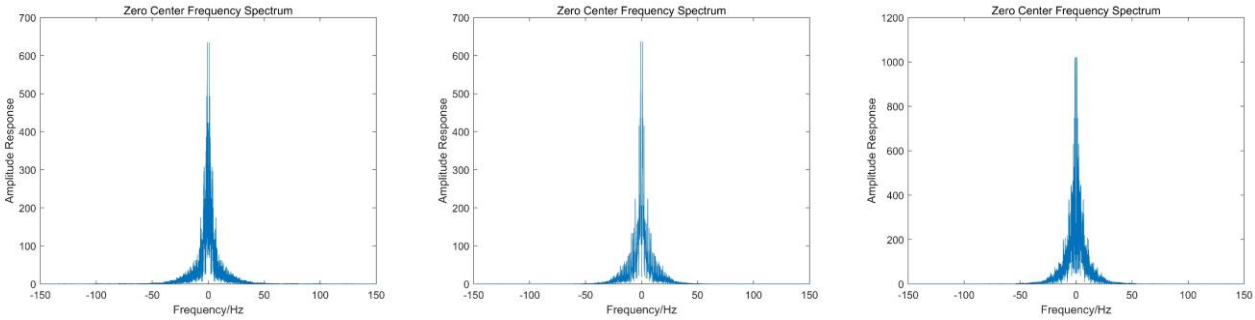


Fig 9. Noisy signal zero-center DFT

Also, we can acquire power spectrum from amplitude spectrum.

$$Power\ Spectrum = \frac{|Amplitude\ Spectrum|^2}{N}$$

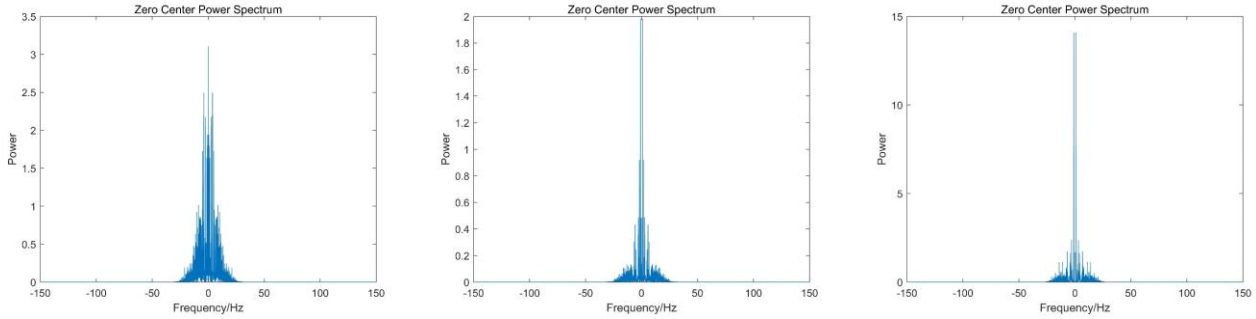


Fig 10. Normal signal zero-center power spectrum

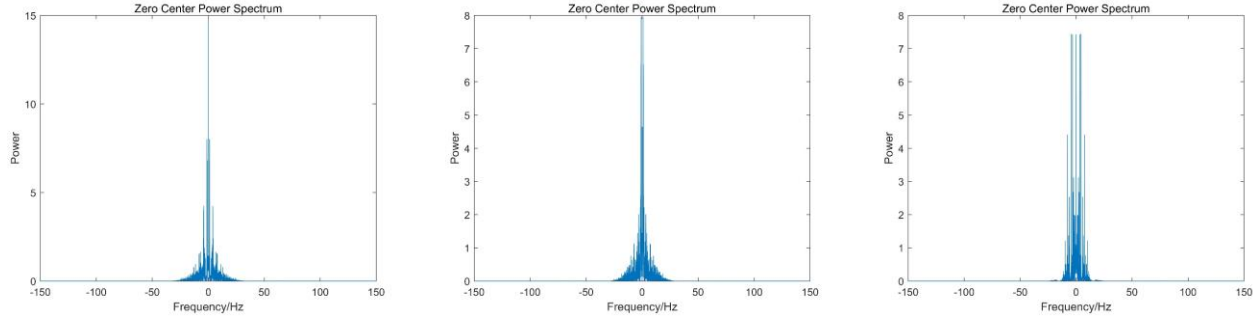


Fig 11. Afib signal zero-center power spectrum

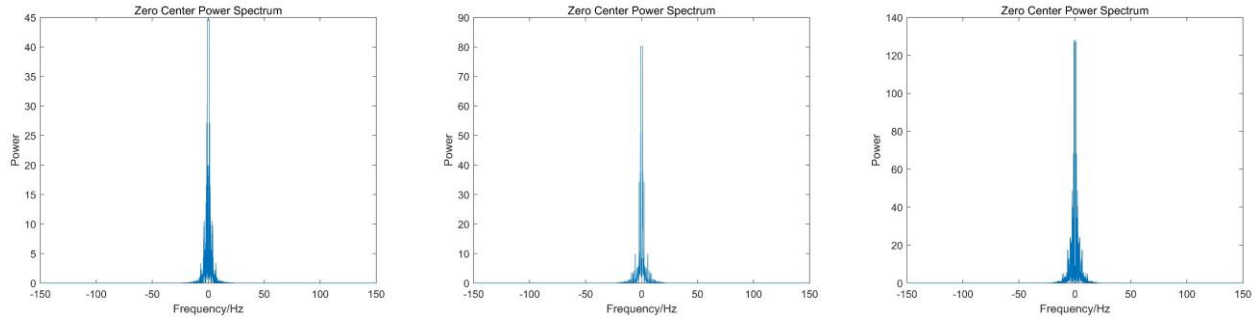


Fig 12. Noisy signal zero-center power spectrum

We take A00001 and A00004 as examples to observe the amplitude spectrum in details.

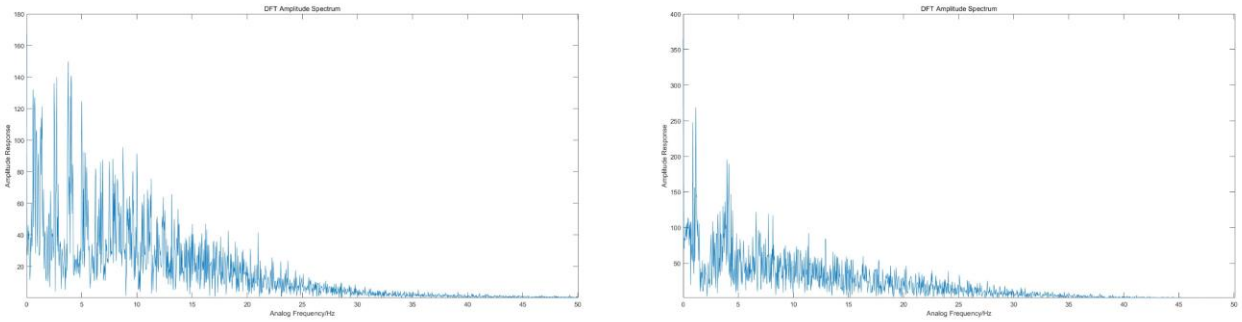


Fig 13. Details of DFT amplitude spectrum (left: A00001; right: A00004)

From above figures we will be aware of the difference of spectrum distribution between normal signals and Afib signals. Noisy signals, compared with normal ones and Afib ones, have a much sharper spectrum, and the spectrum indicates higher energy in the lower band, which may be relevant to the noise.

4.3 Digital filter design

The design of digital filter is to remove the noise mixed in ECG signals. According to the acquirement of ECG signals, relevant noise source is listed below:

·Baseline wandering: 0.1 – 3 Hz.

·Muscle: 5 - 50 Hz.

·External electrical: 50 or 60 Hz (A/C mains or line frequency).

·Other electrical: typically > 10 Hz (muscle stimulators, strong magnetic fields, pacemakers with impedance monitoring).

According to the power spectrum we acquired above, we can see that the energy of ECG is concentrated between 0 – 25 Hz. With baseline wandering taken consideration, we need to design a bandpass digital filter to process the signals. The design requirements are described below:

·Stop band frequency: $\omega_{st1} = 3\ Hz;\ \omega_{st2} = 28\ Hz$

·Pass band frequency: $\omega_{p1} = 6\ Hz;\ \omega_{p2} = 25\ Hz$

·Transition band width: $\omega_t = 3\ Hz$

·3db cutoff frequency: $\omega_{c1} = 4.5\ Hz;\ \omega_{c2} = 26.5\ Hz$

·Passband ripple: 0.01

·Stopband ripple: 0.05

·Passband magnification: 1.0

·Stopband magnification: 0.0

According to the requirements, we design a FIR digital filter with Kaiser window. Compared with IIR filters, FIR filters can avoid non-linear phase, which is important to maintain the shape of waveform. We invoke function *kaiserord* to design the filter. The amplitude response and phase response are plotted below:

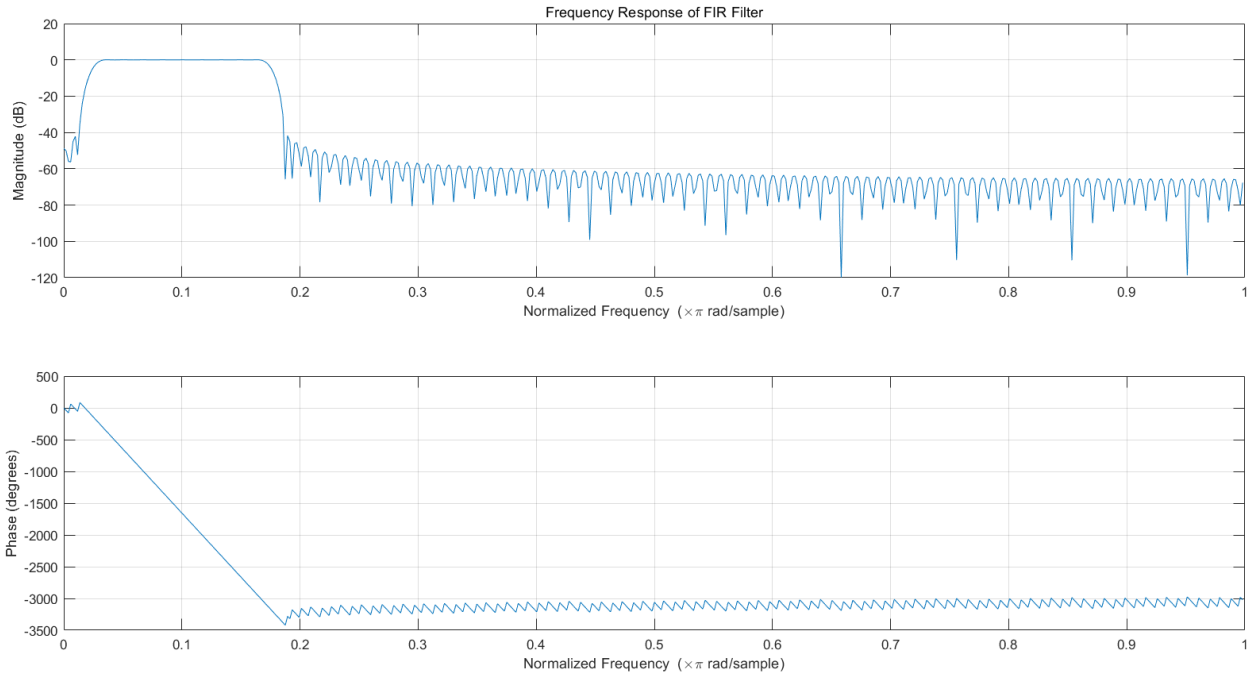


Fig 14. Frequency response of FIR filter

It should be noted that FIR filter will induce delay in time domain, which shall be eliminated to ensure that the waveform is not distorted.

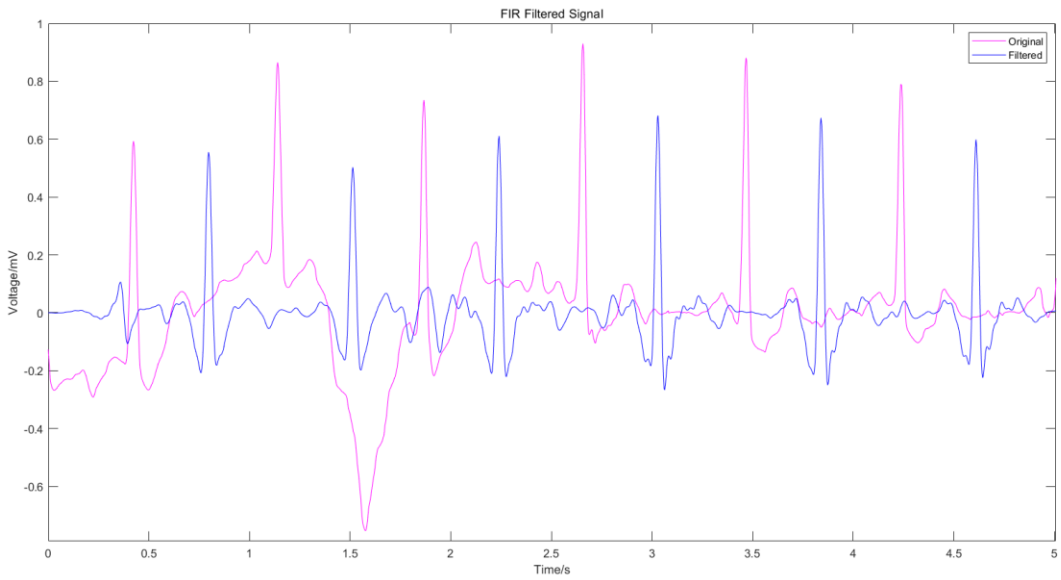


Fig 15. Delay in time domain

After the design of FIR filter, we apply the filter to original signals and remove delay in time domain. Filtered noisy signals are plotted below.

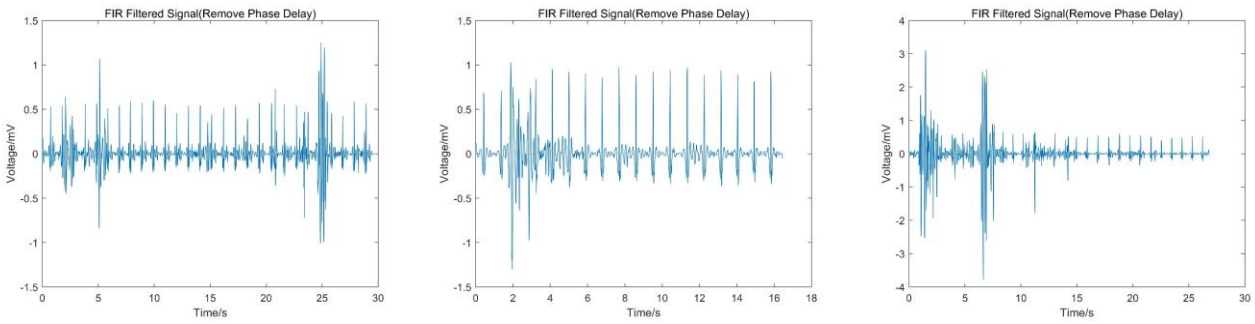


Fig 16. Filtered noisy signals

Compared with original ones, we can see that the signal quality has been improved significantly. Some of the patterns are recovered through bandpass filtering, and baseline wandering has been completely removed. Still, some severe noise exists in the signal after filtering, which indicates that certain noise's frequency overlaps with ECG's, and cannot be removed without damage to useful patterns.

Also, we apply bandpass filter to normal and Afib signals to observe the effects. Filtered normal signals and Afib signals are plotted below.

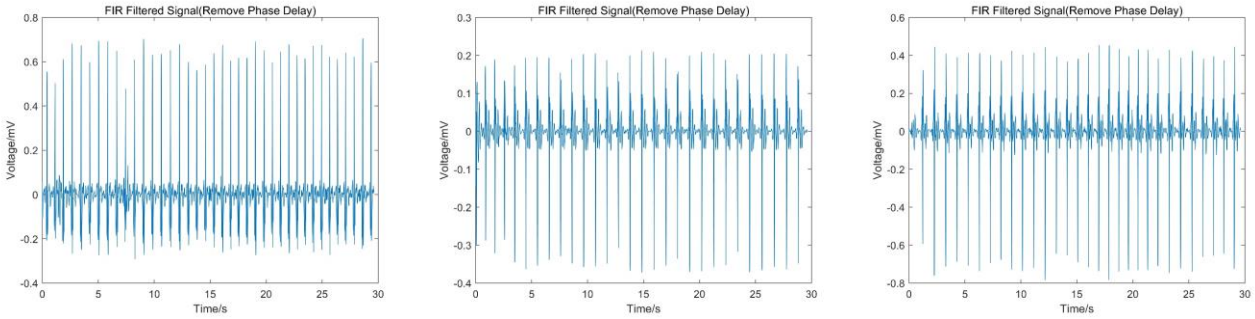


Fig 17. Filtered normal signals

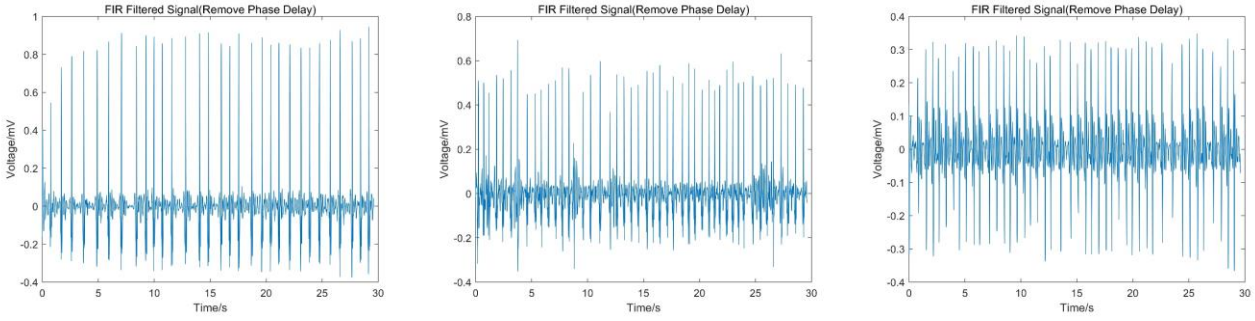


Fig 18. Filtered Afib signals

It is obvious that bandpass filtering is of benefit for normal and Afib signals as well. Baseline wandering is removed after filtering and high-frequency noise is also suppressed. Filtered normal signals and Afib signals can be further utilized for calculating heart rate and extracting features for classification.

4.4 Heart rate calculation

4.4.1 Average heart rate calculation

ECG signals reflect bioelectric changes in the process of generation, conduction and recovery of cardiac excitation. In each cardiac cycle, ECG signal contains P-wave, QRS-wave and T-wave. A typical ECG template is plotted below.

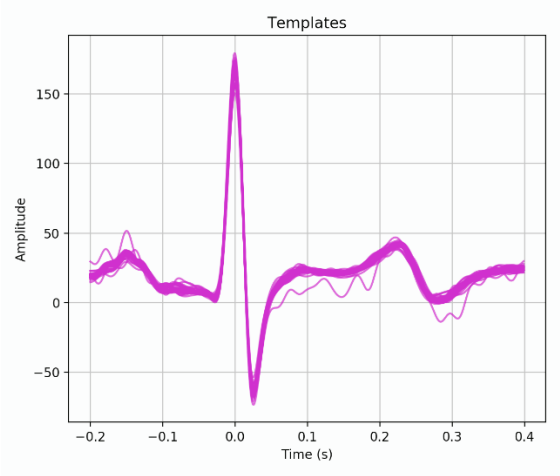


Fig 19. Template of ECG

To calculate average heart rate, we need to find all the R-peaks in the signal's duration. Assume the number of R-peaks M, signal length N (start from 0), sampling frequency f_s, we have

$$Average\ Heart\ Rate = 60\frac{Mf_s}{N-1}\ bpm$$

We take A00001 and A00009 as examples for calculating heart rate. First, we use Python library *biosppy* to perform R-peak segmentation. After segmentation, function *biosppy.signals.ecg.hamilton_segmenter* returns the indices of R-peaks in the duration. The segmented R-peaks are plotted below.

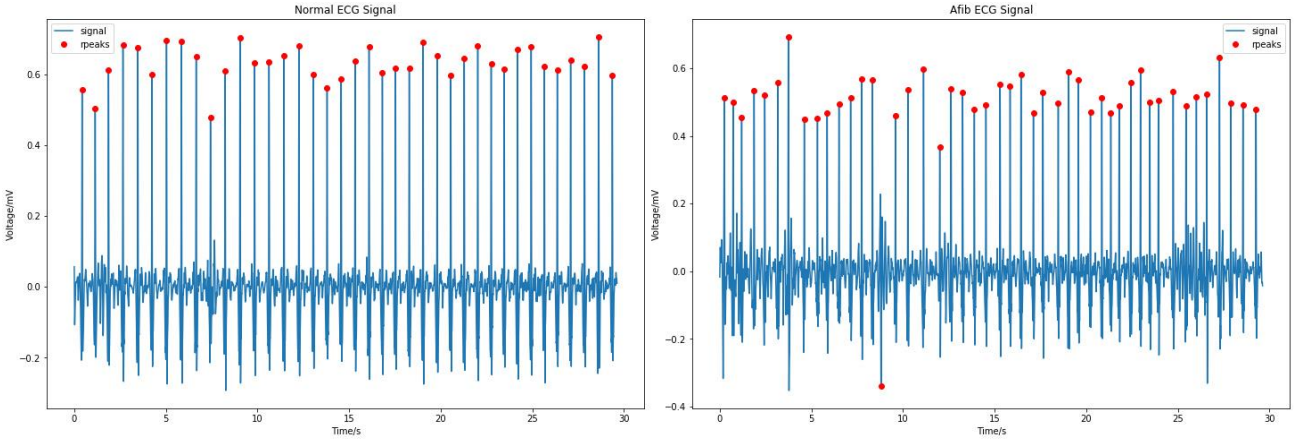


Fig 20. R-peak detection (left: A00001; right: A00009)

After segmentation we can calculate heart rate from above equation. Heart rate of A00001 is 78.98 *bpm* and heart rate of A00009 is 95.18 *bpm*.

4.4.2 Instantaneous heart rate calculation

We can differentiate the indices of R-peaks to acquire the interval between every two heart beats. Thus, we can estimate the instantaneous heart rate by this equation:

$$Instantaneous\ Heart\ Rate = \frac{60 f_s}{\Delta\,(R - peak\ indices)}$$

Then we can plot the heart rate diagram of certain participant in the period.

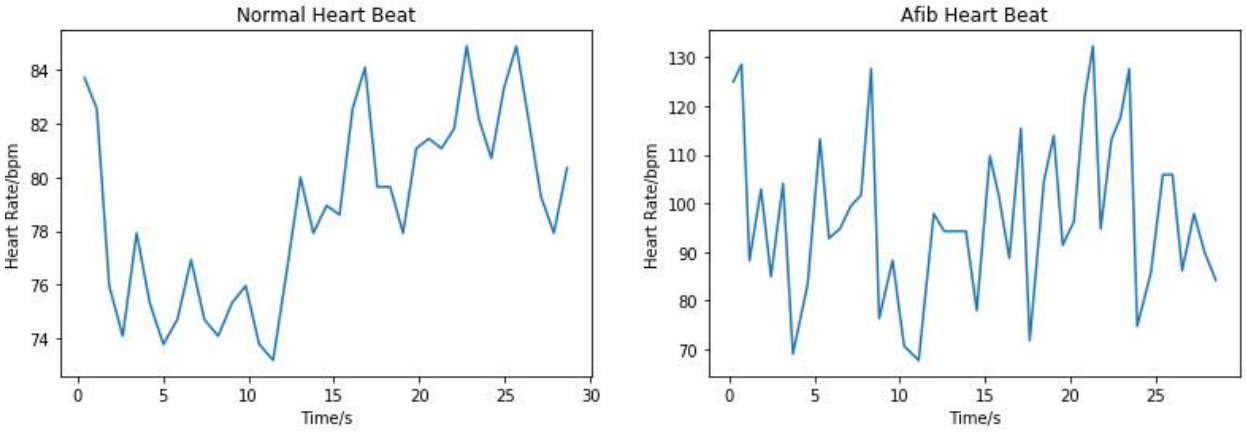


Fig 21. Instantaneous heart rate (left: A00001; right: A00009)

4.5 Binary Classification

4.5.1 Feature engineering

The performance of machine learning task is highly affected by feature engineering. In the task we modified open source code based on MIT license, which was put forward by Sebastian D. Goodfellow. We remove the filtering part of the source code since in our task filtering has been previously performed in Matlab. Also, we add R-peak related features (mean of diff R-peak indices, std of diff R-peak indices) and heart rate related features (average heart rate, std of instantaneous heart rate) to the extracted features since these features are important to recognize Afib signals in time domain.

Before extracting features, we choose 500 ECG signals which include 250 normal ones and 250 Afib ones. The number of both positive class and negative class is equal to make sure the task a balanced binary classification, which will save efforts in modifying classification algorithms. Then feature engineering is performed on chosen participants to extract useful features. After feature engineering, we acquire a feature diagram which contains 52 groups of features.

4.5.2 Classification

We perform classification with Python library *scikit-learn*. Three types of classification algorithms including support vector machine (SVM), logistic regression (LR) and random forest (RF) are evaluated in the task.

First, all the statistics are normalized by z-score method to suppress the effect of unbalanced range of different features:

$$z = \frac{x - \mu}{\sigma}$$

where μ means the mean value of the feature, σ means the variance of the feature.

After normalization, we split the dataset into training set and test set with a ratio of 4:1. For SVM and RF, a 5-fold cross validation is performed to find the best parameters of the classifier. Chosen metrics for binary classification includes accuracy (ACC), specificity (SPE), sensitivity (SEN), receiver operating characteristic curve (ROC) and area under curve (AUC).

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

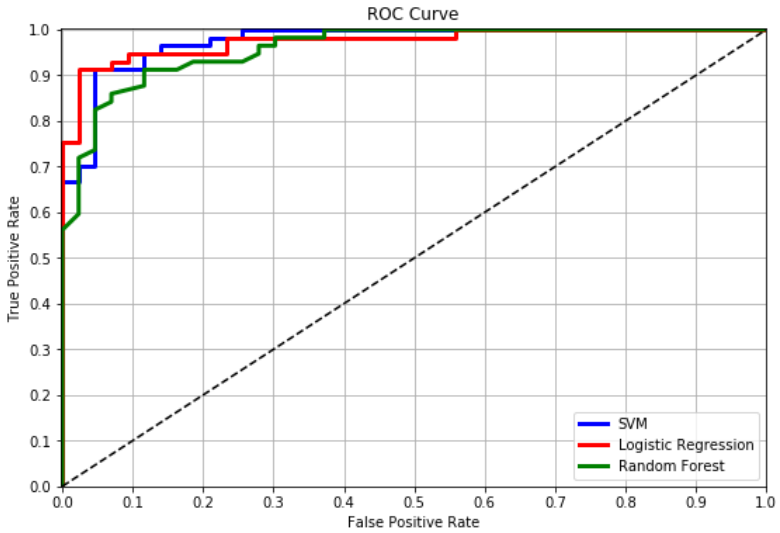
$$SPE = \frac{TN}{TN + FP}$$

$$SEN = \frac{TP}{TP + FN}$$

where *TP* means true positive, *TN* means true negative, *FP* means false positive, *FN* means false negative.

Classification results are printed below.

Classifier	ACC	SPE	SEN	AUC
SVM	0.920	0.953	0.895	0.975
Logistic Regression	0.920	0.977	0.877	0.976
Random Forest	0.890	0.884	0.895	0.960



IV. Discussion

Biomedical signal processing is the fundamental of many tasks, including physiological information calculation, machine learning tasks such as classification, regression and segmentation. The basic principle of biomedical signal processing is to observe and evaluate signals in both time domain and frequency domain. Filtering is significant for processing signals to explore hidden patterns of signals.

However, compared with ordinary signals, biomedical signals have certain features: low amplitude, high noise, low frequency and completely random. The features make biomedical signal processing a hard task because it is difficult to process signals without damaging internal patterns of them. Thus, developing new processing algorithms is of great significance. In the task, the effect of filtering of noisy signals is limited because the noise overlaps signals in frequency domain. Such conditions cannot be avoided, thus making the development of robust algorithms rather important.

For normal and Afib signals, filtering is also of great significance. Although the noise level of those signals is tolerable, filtering is still of great importance to remove baseline wandering and high frequency noise. From the task it can be noticed that the patterns of signals are strengthened after filtering, which can be of great benefit for further applications.

Feature engineering is crucial for most of the machine learning tasks. We can see that popular classification algorithms perform equally with the same feature engineering. The result of classification is relatively good because of good data quantity and quality, and also successful feature engineering.

In conclusion, the whole task provides the basic procedures and the holistic perspective of biomedical signal processing, which is rather inspiring for us.

Reference:

1. Goodfellow, S. D., A. Goodwin, R. Greer, P. C. Laussen, M. Mazwi, and D. Eytan (2018), Atrial fibrillation classification using step-by-step machine learning, Biomed. Phys. Eng. Express, 4, 045005. DOI: 10.1088/2057-1976/aabef4

2. Carreiras C, Alves AP, Lourenço A, Canento F, Silva H, Fred A, et al(2015), BioSPPy - Biosignal Processing in Python, https://github.com/PIA-Group/BioSPPy/

Supporting Materials

1. folder *figure*

This folder contains all the processed signal plot.

2. demo.mlx

Matlab 2018a real time script; demonstrates the procedure of processing a signal in Matlab.

3. filtering.m

Plot filtered signals.

4. main.m

Plot time-domain signals, DFT and power spectrum.

5. prepare_data.m

Filter signals and save as Matlab files for further usage (heart rate calculation and machine learning).

6. rpeak_detection&heart_rate_calculation.ipynb

Jupyter Notebook file; detect R-peaks and calculate heart rate.

7. folder *ML*

The folder contains all the scripts for machine learning task.

7.1 folder *data*

The folder contains data needed, including filtered signals and features; also contains results, including classifier performance and ROC curve.

7.2 folder *features*

Contains scripts for feature engineering. Open sourced by Sebastian D. Goodfellow; modified to suit the task.

7.3 folder *utils*

Contains scripts for feature engineering. Open sourced by Sebastian D. Goodfellow.

7.4 machine_learning.ipynb

Jupyter Notebook file; procedure of machine learning.

7.5 feature_engineering.py

I/O script for extracting features.

7.6 evaluate_model.py

Scripts for evaluate model; calculate necessary metrics.