Sheffield Hallam University

Atrial fibrillation detection service validation tool

FAUST, Oliver <http://orcid.org/0000-0002-3979-4077>, KAREEM, M and LEI, Ningrong <http://orcid.org/0000-0003-0935-9426>

Available from Sheffield Hallam University Research Archive (SHURA) at:

https://shura.shu.ac.uk/30086/

This document is the Published Version [VoR]

Citation:

FAUST, Oliver, KAREEM, M and LEI, Ningrong (2021). Atrial fibrillation detection service validation tool. Software Impacts, 10: 100117. [Article]

Copyright and re-use policy

See http://shura.shu.ac.uk/information.html

Sheffield Hallam University Research Archive
http://shura.shu.ac.uk

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Atrial fibrillation detection service validation tool

Oliver Faust[a,*], Murtadha Kareem[b], Ningrong Lei[a]

[a] Department of Engineering and Mathematics, Sheffield Hallam University, Howard St, Sheffield S1 1WB, UK
[b] Materials and Engineering Research Institute, Sheffield Hallam University, Howard St, Sheffield S1 1WB, UK

ARTICLE INFO

Keywords:
Algorithm validation
Deep learning
Parallel processing
Clinical data processing

ABSTRACT

We developed a software tool to validate a deep learning algorithm for an atrial fibrillation detection service with heart rate data from a clinical study. The deep learning algorithm analyses the measurement data and establishes an estimated atrial fibrillation probability for each heartbeat. The software tool displays both data and deep learning analysis results. Furthermore, the graphical user interface can be used by medical experts to detect atrial fibrillation periods in the data and establish a reference result which will be treated as ground truth in subsequent result analysis steps. Once both deep learning and expert results are available, a confusion matrix is produced and the algorithm performance is validated by establishing accuracy, sensitivity, specificity, and f1-score. The software tool was created in Python and the software incorporated a graphical user interface as well as functional elements for data display and deep learning. To establish the required functionality, we used three different parallel processing methods for: (1) user interface processing, (2) data handling, and (3) deep learning. This highlights the need for parallel processing methods even for projects with a low or mid-range complexity. We have learned that the functionality of individual components can be expressed elegantly in Python. However, the lack of parallel debugging support makes it rather difficult to integrate functional components to establish a working solution.

Code metadata

Current Code version	0.1
	The clinical study, the measurement of which are assessed with the tool, is still ongoing. Therefore, it is expected that the tool is not feature complete.
Permanent link to code/repository used of this code version	https://github.com/SoftwareImpacts/SIMPAC-2021-82
Permanent link to Reproducible Capsule	
Legal Code License	GNU General Public License v3.0
Code Versioning system used	SVN
Software Code Language used	Python 3.7.7
Compilation requirements, Operating environments & dependencies	Linux (Ubuntu), Windows
If available Link to developer documentation/manual	None
Support email for questions	oliver.faust@gmail.com

Software metadata

Current software version	0.1
Permanent link to executables of this version	There is no executable for this program. As such, the program has a very narrow scope, therefore producing an executable was not a requirement. However, we tried to generate an executable for distribution and found that the dependencies were too complex. Especially the deep learning modules and QT were incompatible with standard methods for building executables.
Permanent link to Reproducible Capsule	
Legal Software License	GNU General Public License v3.0
Computing platform/Operating System	Microsoft Windows, Unix-like
Installation requirements & dependencies	Python 3.7.7
If available Link to user manual - if formally published include a reference to the publication in the reference list	
Support email for questions	

* Corresponding author.
E-mail address: oliver.faust@gmail.com (O. Faust).

https://doi.org/10.1016/j.simpa.2021.100117
Received 1 August 2021; Received in revised form 6 August 2021; Accepted 8 August 2021

2665-9638/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

The need for the proposed software tool is linked to stroke, which is the second leading cause for mortality and the third leading cause of disability worldwide [1,2]. Prevalence in the general population and severities of outcome are the main drivers behind that statistic. Stroke refers to the detrimental processes that occur when brain tissue is deprived of oxygen [3]. An Ischemic stroke occurs when debris, such as blood clots and plaque, blocks an artery. The occurrence of debris is linked to the fluid dynamics in the cardiovascular system. The way in which the heart pumps blood has a major impact on the fluid dynamics. Hence, the heart rhythm is linked to the occurrence of debris which influences the stroke probability. Studies have found that the heart rhythm irregularity known as atrial fibrillation is particularly dangerous because it is rather common, and it will increase the stroke risk fivefold [4,5]. Unfortunately, some forms of atrial fibrillation are difficult to detect because the signature rhythm irregularity is not present all the time. The national health service in England estimates that only 79% of all atrial fibrillation cases are detected [6]. Technical solutions are required which extend the observation duration and thereby improve the detection rate.

One way of detecting atrial fibrillation is to monitor the heart rate of a patient and analyze the resulting signal for signs of heart rhythm irregularity [7]. We have trained a long-short term memory deep learning algorithm with data from the PhysioNet atrial fibrillation database to create a heart rhythm irregularity detection model [4]. That model can detect atrial fibrillation with an accuracy of 99%. Encouraged by the excellent detection capability [4], we designed an atrial fibrillation detection service based on internet of medical things technology [6]. The idea is that the measured signals flow from the patient to a cloud server where our algorithm analyses the signal in real time [8]. Hybrid diagnosis support will ensure proper human validation during the diagnostic process [9–11]. Having wireless connectivity and real time analysis extends the observation duration indefinitely and thereby presents a viable solution to the atrial fibrillation detection problem [8]. However, before the service can be deployed, it is necessary to validate the deep learning algorithm in a clinical environment.

To address the validation problem, we have created a software tool which tests the deep learning functionality of the atrial fibrillation detection service. That software automates the comparison between cardiologist and deep learning results. For these comparisons, we treat the cardiologist results as ground truth against which we benchmark the deep learning results. Each heartbeat is labeled by both cardiologist and machine algorithm as either AF or non-AF. All the labels are summarized in a confusion matrix and abstract performance measures, such as accuracy, sensitivity, and specificity are calculated. Furthermore, the graphical capabilities of the software tool foster detailed discussions between cardiologists, computer scientists, and biomedical engineers which might help to improve the proposed atrial fibrillation detection service.

The remainder of this manuscript introduces the software tool by following the systems engineering methodology. The next section details the software requirements. Section 3 refines these requirements into a specification. The implementation discusses some aspects of the coding. The last section provides conclusion, limitation, and future work.

2. Requirements

The following requirements describe what software we should build [12]. The software tool should automate the comparison between human expert and deep learning results. The measurements should be processed offline with the previously developed deep learning algorithm. The software should allow human experts to input the ground truth analysis results. The output of the program should be a confusion matrix and classification quality measures.

3. Specification

The requirements are refined into a specification which defines how we build the software [13]. The software takes excel sheets as input. These excel sheets are produced by the Lifeguard server from Isansys and they contain heart rate and electrocardiogram signals amongst other information. The heart rate signals are analyzed with the long-short term memory deep learning algorithm, which produces an estimated AF probability score for each heartbeat [4]. The analysis results are displayed alongside the signal data in two dimensional graphs. Navigation through these graphs is established with crosshair functionality. These graphs can be used to review and annotate the electrocardiogram signal with regions of atrial fibrillation [14,15]. The annotated regions are treated as ground truth with which the deep learning result is compared. To be specific, every beat that falls within the annotated region is treated as AF and every beat outside the region is non-AF [16]. A threshold is used to generate regions of estimated AF. This is done by comparing the threshold value with the estimated AF probability for each heartbeat. Whenever the estimated AF probability value is larger than the threshold, that beat belongs to a region of estimated AF. As a result, each beat has two labels: one from a human expert and one from the deep learning algorithm. Based on these labels a 2×2 confusion matrix is established. The matrix elements are used to calculate the performance measures of accuracy, sensitivity, specificity, and f1-score. These performance results, together with the estimated AF probability as well as the expert and algorithmic regions are saved in a separate excel file.

4. Implementation

The implementation was a meandering journey between learning the Python language and establishing the specified functionality. Despite the relative inexperience with the language itself, the need for parallel processing became apparent early in the implementation cycle. To start, the inference functionality of Kearas, which utilizes our deep learning model, incorporates parallel processing to establish the estimated atrial fibrillation probability [17]. Fortunately, this functionality is very well abstracted and indeed hidden from the user. Engaging with parallel processing libraries was required to realize a speedup for the signal display processing. We have used *pyscp* to compose electrocardiogram, heart rate, and estimated atrial fibrillation probability data vectors in parallel. Composing these vectors and the inference processing has high and very high computational complexity, respectively. This translates into waiting times for the program user. A progress spinner was implemented which indicates the processing of a potentially long task. This required us to use the QT multithreading functionality.

We have successfully established the specified functionality with three different parallel processing methods. However, the lack of debug support for parallel processing in the Python development environment Spyder made that task unnecessarily hard. At times we resorted to trace messages and sometimes we bypassed the Qt multithreading functionality to inspect variables in code which is normally executed as a Qt thread. Furthermore, there is also some scope for formalizing the design and standardizing both code as well as file structure. This might lead to improved code quality.

5. Impact review

The atrial fibrillation detection service validation tool automates the comparison between cardiologist and deep learning analysis results [3]. Furthermore, relevant signal graphs allow the users to inspect the analysis results which might lead to a deeper understanding of artificial intelligence for atrial fibrillation detection. To be specific, for medical professionals it is important to visualize the deep learning processing, because they want to establish what mistakes are happening and the extend of overreporting and underreporting of events. Based on this visualization, we can start a conversation on what action to take as a result of a specific scenario. For example, we might be able to answer questions like: How much automatically detected atrial fibrillation justifies interventions with potentially life-threatening side effects, such as anticoagulation. Currently, this is where we draw the line between machine and human work. The machine provides an estimated atrial fibrillation probability over time and cardiologist must interpret that result. The interpretation should be done by fusing the result information with knowledge about the patient to reach a diagnosis.

Establishing a hybrid environment where humans work with machine algorithms is an important goal for future work [9]. The current atrial fibrillation detection service validation tool can only serve as an initial attempt with which we can study interaction patterns. These patterns might indicate a direction for further automatization. Currently, we are thinking about rules to establish alarm situations. Understanding the estimated atrial fibrillation probability signal shape for a treatable case might lead to the automated generation of alarm messages. For example, an alarm message is sent when the estimated atrial fibrillation probability is above a 50% threshold for more than 5 min within one hour. Calculating that and disseminating the alarm message is straight forward, but significantly more research is needed to establish a useful amount of alarm cases. The focus on establishing alarm message conditions might seem like a minor point, but this is what lies at the heart of all internet of medical things devices that provide diagnosis support by measuring, distributing, and analyzing patient data in real time. These systems are capable to extend the observation duration indefinitely which holds the promise of detecting diseases earlier and that detection is largely independent from whether there are long asymptomatic episodes [6]. Having the long observation duration together with the alarm functionality is likely to improve outcomes for patients through an early diagnosis which will lead to less intrusive interventions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by Grow MedTech (Research England) as part of a proof of feasibilitystudy, grant number PoF000099.

References

[1] P.A. Wolf, R.D. Abbott, W.B. Kannel, Atrial fibrillation as an independent risk factor for stroke: the Framingham study, Stroke 22 (8) (1991) 983–988.
[2] A.G. Thrift, D.A. Cadilhac, T. Thayabaranathan, G. Howard, V.J. Howard, P.M. Rothwell, G.A. Donnan, Global stroke statistics, Int. J. Stroke 9 (1) (2014) 6–18.
[3] N. Lei, M. Kareem, S.K. Moon, E.J. Ciaccio, U.R. Acharya, O. Faust, Hybrid decision support to monitor atrial fibrillation for stroke prevention, Int. J. Environ. Res. Public Health 18 (2) (2021) 813.
[4] O. Faust, A. Shenfield, M. Kareem, T.R. San, H. Fujita, U.R. Acharya, Automated detection of atrial fibrillation using long short-term memory network with RR interval signals, Comput. Biol. Med. 102 (2018) 327–335.
[5] O. Faust, U.R. Acharya, Automated classification of five arrhythmias and normal sinus rhythm based on RR interval signals, Expert Syst. Appl. 181 (2021) 115031.
[6] M. Kareem, N. Lei, A. Ali, E.J. Ciaccio, U.R. Acharya, O. Faust, A review of patient-led data acquisition for atrial fibrillation detection to prevent stroke, Biomed. Signal Process. Control 69 (2021) 102818.
[7] O. Faust, Y. Hagiwara, T.J. Hong, O.S. Lih, U.R. Acharya, Deep learning for healthcare applications based on physiological signals: A review, Comput. Methods Programs Biomed. 161 (2017) 1–13.
[8] O. Faust, W. Yu, U.R. Acharya, The role of real-time in biomedical science: A meta-analysis on computational complexity, delay and speedup, Comput. Biol. Med. 58 (2015) 73–84.
[9] M. Kareem, O. Faust, Establishing the safety of a smart heart health monitoring service through validation, 2019, pp. 6089–6091.
[10] O. Faust, E.J. Ciaccio, A. Majid, U.R. Acharya, Improving the safety of atrial fibrillation monitoring systems through human verification, Saf. Sci. 118 (2019) 881–886.
[11] O. Faust, U.R. Acharya, B.H. Sputh, T. Tamura, Design of a fault-tolerant decision-making system for biomedical applications, Comput. Methods Biomech. Biomed. Eng. 16 (7) (2013) 725–735.
[12] O. Faust, B.H. Sputh, U. Acharya, A.R. Allen, A pervasive design strategy for distributed health care systems, Open Med. Imaging J. 2 (2008) 58–69.
[13] O. Faust, R. Acharya, B.H. Sputh, L.C. Min, Systems engineering principles for the design of biomedical signal processing systems, Comput. Methods Programs Biomed. 102 (3) (2011) 267–276.
[14] O. Faust, R.J. Martis, L. Min, G.L.Z. Zhong, W. Yu, Cardiac arrhythmia classification using electrocardiogram, J. Med. Imaging Health Inf. 3 (2013) 448.
[15] A. Mohsin, O. Faust, Automated characterization of cardiovascular diseases using wavelet transform features extracted from ECG signals, J. Mech. Med. Biol. 19 (01) (2019) 1940009.
[16] U.R. Acharya, O. Faust, E.J. Ciaccio, J.E.W. Koh, S.L. Oh, R. San Tan, H. Garan, Application of nonlinear methods to discriminate fractionated electrograms in paroxysmal versus persistent atrial fibrillation, Comput. Methods Programs Biomed. 175 (2019) 163–178.
[17] O. Faust, M. Kareem, A. Shenfield, A. Ali, U.R. Acharya, Validating the robustness of an internet of things based atrial fibrillation detection system, Pattern Recognit. Lett. 133 (2020) 55–61.