Cloud Service for Electrocardiogram Registration for Using in Authentication

Yurii Luhovskyi [1]

[1] *The Institute Of Mathematical Machines and Systems Problems National Academy Of Science Of Ukraine (IMMSP NASU), 42, Academician Glushkov Avenue, Kyiv, 03187, Ukraine*

Abstract

Recently a need for distance consultation between a patient and a doctor grew rapidly. It caused telemedicine to develop faster. New technologies and devices have been created that have improved telemedicine and extended the area of application. E.g. home monitoring. It requires specific devices to read health indicators and send them to a doctor. One of the indicators is electrocardiogram. A doctor who received the electrocardiogram must be sure that it belongs to the patient. For this purpose, the service described in the paper has been created for. It was created to perform biometric authentication using heart beats. This technology can be used at any place where authentication is required.

The developed product is a web service that is built on microservices and deployed to the cloud. It has an architecture that can be used in a production environment. The approaches, algorithms and methods used in the web services are described. The paper provides information about the composition of microservice architecture and the responsibility of each microservice. The communications between them are described as well.

The evaluation of web server performance has been done. It was based on the total time required to register the whole set of input ECGs. At the moment the web service only performs the registration. The artifacts of web service are used for authentication by another tool.

The experiment results show that the productivity of the developed architecture was improved by executing part of the work concurrently. However, with the allocation of more computing resources, productivity started dropping down after a specific configuration setup. The results showed that the issue is caused by the machine learning library. The measured performance confirmed Amdahl's law.

At the end provided suggestions for improving the current architecture setup and further web service development.

Keywords

Electrocardiogram, authentication, microservice architecture, machine learning, Amdahl's law

1. Introduction

Telemedicine is being developed much faster and more intensively in recent years. After the pandemic of Covid-19 a need for distance consultations and monitoring rapidly grew for medical institutions. It was the period when a lot of medical facilities wanted to go for distance consultation as soon as possible. This need forced the development of telemedicine. New technologies with devices

[1] 13th International Scientific and Practical Conference from Programming UkrPROGP'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: roop56@ukr.net (A. 1)
ORCID: 0000-0002-0195-2770 (A. 1)

to them were introduced. Nowadays telemedicine is still developing fast and each month service becomes better.

One of the features that telemedicine suggests is the remote monitoring of a patient. The idea is to check the state of health. Periodically some health indicators are collected with the help of devices and sent to a doctor. We are interested only in one indicator which is an electrocardiogram (ECG). There are portable electrocardiographs that patients can use at home. After recording an ECG the patient sends it to a doctor and the doctor should be sure that the received ECG belongs to their patient. This criterion is important for insurance companies that need to pay a patient in case a doctor prescribes medicines. In this case, the technology of checking the electrocardiogram for belonging to a specific patient comes in handy. This approach is used in information technology and is named authentication. The idea of using the same ECG for authentication and by a doctor makes an impossible scenario when a doctor analyzes someone else's ECG.

This type of authentication is biometric since the heartbeat is used. It can be used not only in telemedicine but everywhere where authentication is needed, e.g. to provide access to the data or allow entrance to some building or confirm a transaction. That means that the use of ECG authentication can be widely used which creates additional demand.

The big difference that authentication by ECG has due to other biometrics methods is the authentication will be successful only if a person is alive. In some cases, this could be a crucial demand. Moreover, to fake an ECG signal is too hard and devices to read ECGs are not expensive. Taking into account these advantages authentication by ECG is safer and more available than some other methods like authentication by fingerprints or iris recognition.

It’s good to have a project that can provide such services. The paper describes the web service that boosts authentication technology. The authentication happens in two steps in the suggested technology. First, the ECG of the person is registered in the system. The second step is authentication. Authentication happens by comparing registered and received ECGs. The paper describes a developed cloud service that does preparation work for authentication. It registers ECGs. The main idea of the service is to show that it can register ECGs and because it is based in the cloud can perform calculations faster than a sequential program. Besides, the web service needs to have the ability to work with a big amount of ECGs since the population on Earth is more than 8 billion which means the variety of input data is great and it needs to be taken into account.

Based on these demands the cloud service has been developed. It can register new ECGs and deal with its big amount. The paper shows the performance of the service based on the total time required to register 30 ECGs. The received results showed that the service works but needs to be improved.

The algorithms and methods that are used in the service are described as well. Mentioned from which components the service consists of and how communication and data transferring happens inside it. Also, described which artifacts are created by the service.

2. Methods and algorithms

Authentication by ECGs is a relatively new technology that is still in the research, developing and testing state. There are a lot of papers in this research field. Commonly, they describe all processes required to perform authentication. Usually, papers contain data descriptions and methods that are used to process ECG signals. They describe the ways of selecting features from ECGs and how machine learning or other comparison methods are used to compare these features. The results of the comparison are used to determine if ECGs belong to the same person. In the end, the summary results show the efficiency of the chosen method.

Each research uses specific sets of methods and algorithms that provide authentication. For processing input ECGs signals researchers use Kalman filter [1], infinite impulse response filter [1], band pass filter [3, 4] and others. They use different methods to extract features. There are two types of features: fiducial and non-fiducial. Fiducial features are values that can be measured, e.g. cardiac cycle time, peak value, time between peaks. Non-fiducial features are presented as parameters or coefficients that in couple with some methods describe the ECG signal. These methods include discrete cosine transform [5, 6], autocorrelation [7], wavelet transform [6, 7, 15]. Moreover, there are approaches that use machine learning to learn models and classify using pure ECG signals [3, 7, 9,

10]. The last part is to identify if the passed ECG for authentication is the same as the original one. To compare ECGs features used different machine learning technologies [3, 4, 7, 9, 10, 11, 12, 13, 14, 15] or methods that can provide a value that represent the similarity of compared ECGs, e.g. Euclidean distance, hamming distance, dynamic time warping [5, 15, 16, 17, 18, 19, 20]

The web service described in the paper uses algorithms and approaches developed by researchers from IMMSP which are presented in the papers [21, 22, 23, 24, 25, 26, 27]. The processing consists of extracting the QRS complex from a cardiac cycle, extracting features from QRS and using them as input for machine learning to train a neural model. The trained model is used to compare the original ECG with another one.

The purpose of the technology described in the paper is to provide the possibility to authenticate only one person. There is no goal to recognize a set of ECGs. So the result is relevant only to one person. The person can be authenticated or not.

The input data for the processing is a file with ECG signals that are recorded from 3 leads. The signal is a set of cardiac cycles. At first the QRS complexes are extracted from the signals. Then having 3 QRS complexes provided by each lead, the algorithm builds QRS complexes in 3D phase space. Then QRS complexes with defects are filtered and the representative one is found among others that is the closest to the average. The next step is to build a curve that fits the chosen QRS complex in 3D space the most. This is done by using a 3D spline. To build an approximation spline 4 or more data points are used. Then, data points presented as coordinates are used as input values to train machine learning models. The details can be found in papers [21, 22, 23]. This technology has a patent [27].

Finding features that describe ECGs is the first part of the process. The second part is to use machine learning to create a model and use it to identify similar ECGs. To compare ECGs classification is used. To make classification work two datasets are required. The first one is data points from the initial ECG and the second one is data points from a fake ECG. These initial and fake data points are used to train one neural model. The algorithm requires 10 neural models to perform authentication work. So to create 10 models 10 fake ECGs are used. The authentication result is a summary of the classification outputs of all trained models. Based on the provided result ECG will be authenticated or not.

3. Developed programs

All calculations can be divided mainly into two parts. The first one is ECG processing and the second is machine learning calculations. These parts are logically separated so they were presented as separate programmed components.

When registration or authentication happens ECG is processed in the same way. The output of processing is data points that are presented as coordinates in 3D space. This functionality is related only to ECG processing so it was put into a separate library. It has the name EcgAuth.EcgProcessing. It's good to have a library since ECG processing functionality can be added to any other program easily.

A separate library to work with machine learning was created as well. This library uses open source machine learning library ML.NET under the hood. Developed library is a wrapper which has an interface to work with created ECGs objects. It consumes an array of objects with coordinates and mapped it to objects that ML.NET works with. With the library there is no need to copy mapping logic each time when the ML.NET library is needed in other programs. The library has the name EcgAuth.MachineLearning.Engine.

A program for ECG registration and authentication was developed using these libraries. It is a desktop application with UI. The program confirmed that developed algorithms work. It allows manually registering any number of people with their initial ECGs and a mechanism to authenticate one selected person. Authentication was successful for a person when using one of its ECGs and wasn't if using someone else. The program described in the paper [28].

Another program was created that is able to register a set of ECGs. It uses these two libraries as well. The idea of the program is to measure the performance of developed algorithms. More information is provided in the paper [28]. An input set with 30 ECGs was created for the program.

After the program was executed it became clear which part of the process is time-consuming. Learning neural models took the most time. On average to register one ECG machine learning took 77% of all time. ECG processing consisted of 18%-23% of all time. To register one ECG 10 neural modes were created that explained such a big impact. All calculations were executed sequentially. Total time was 12 min 41 sec. In total the program is good to do research but it will take a lot of time to execute for a big set of data.

4. The architecture of the web service

The developed program that can process a set of ECGs is good to validate the technology and do performance testing. This is all of its advantages. Any technology makes sense only when used in production. Nowadays, to make it real this program needs to work in a cloud so different services can connect to it and use its functionality.

Moving the program into the cloud will provide benefits for researchers as well. The cloud has more computation power, so all executions will take less time. Researchers will use only one program, which makes sure they won't work on the outdated version.

The described functionality of the program was moved to the cloud. It performs registration of ECGs that are used for authentication. Created neural models are compatible with the desktop program that does authentication. Registering ECGs is a more sophisticated functionality than authentication since it requires correctly built architecture. Developing registration is the first step in releasing a complete product.

The program deployed into the cloud is based on microservices. Each microservice has its scope of work. The advantage of microservices is that a highly loaded part of computing can be moved to a microservice and provided with more computing power to it. The microservice will be able to process a couple of requests simultaneously in parallel. In case the computing power is insufficient, microservices can be horizontally scaled by adding more instances to reduce a load.

According to the execution result of the program on a set of 30 ECGs, there are two time-consuming areas in calculations. The first is ECG processing and finding coordinates, the second is learning neural models. These functionalities are presented with libraries EcgAuth.EcgProcessing and EcgAuth.MachineLearning.Engine respectively. These libraries were moved to microservices. It made it possible to receive a couple of requests simultaneously on execution. Because of that, the calculation will be done in parallel and the total time will be reduced. The microservices are named EcgAuth.EcgProcessingApi and EcgAuth.MachineLearning.EngineApi.

To organize work between microservices, EcgAuth.ExperimentRunnerApi microservice has been created. It is also used as an entrance point for the web service. It receives the experiment name. Then according to the name, it searches for input ECGs and sends them to find coordinates for machine learning. When coordinates are provided, it triggers machine learning. This microservice is experiment oriented and can't be used in production.

To organize work with machine learning, the EcgAuth.LearningRunnerApi microservice has been created. It is responsible for downloading coordinates and building 10 pairs of data that machine learning will use. The microservice sends requests to train a neural model. It has a configuration parameter that dedicates the amount of simultaneously learning neural models. E.g. If two models need to learn simultaneously two requests will be sent at once and when first finished the new request will be sent at that moment, so at any point in time two neural models will be learning.

The microservices communicate with HTTP requests. The body of the request contains the information for the receiver. The microservices has cloud storage to keep the required files there. The storage is built using high performance object storage MinIO. It contains input ECG files, found coordinates from ECGs, fake ECG coordinates and learned neural models.

This architecture is useful to make experiments. Executing the program means making an experiment. The idea of the experiment is to have every execution of the program unique where generated artifacts are accessible only in the scope of that experiment. Besides, the experiment's results won't be deleted, so it's possible with time to check previous executions. This approach was developed by creating a specific folders structure where each experiment has its own folder. The

experiment's folder contains input ECG files to start the experiment. During an experiment prepared files for machine learning and created neural models will be uploaded to the folder.

Communication between microservices and storage is presented in Figure 1. Arrows show the direction of HTTP requests and responses.

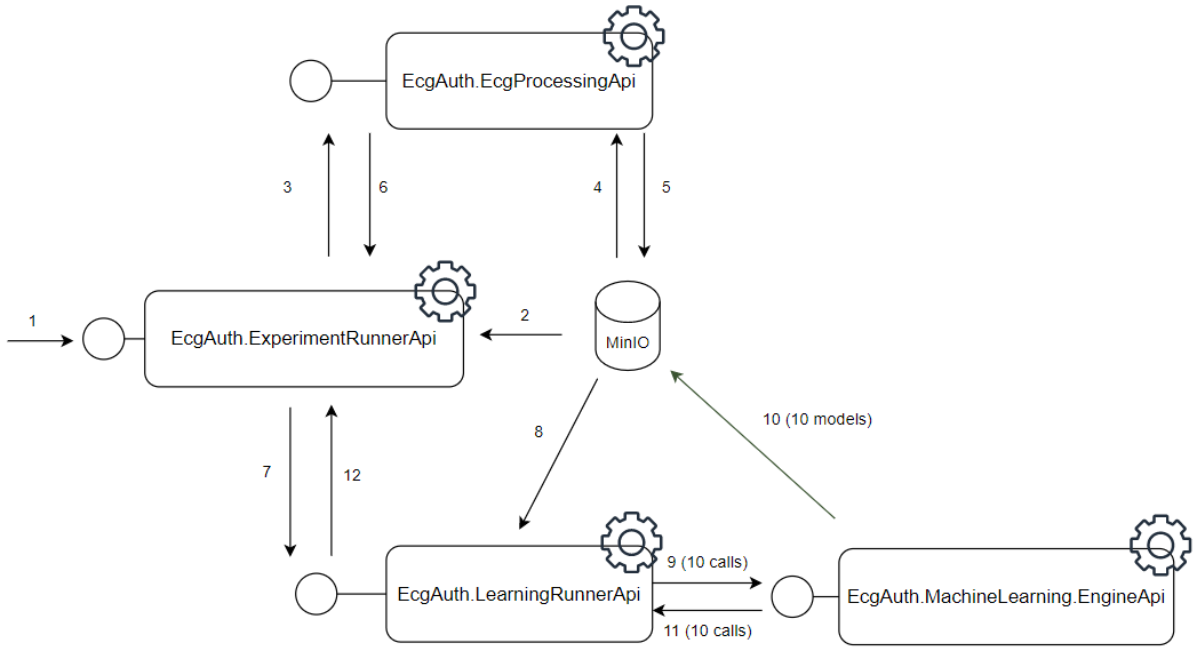

Figure 1. Communication between the microservices and the storage

This is the sequence of requests and responses that are sent and received while an experiment is executed. Besides, data that are moving between requests are described as well.

1. EcgAuth.ExperimentRunnerApi receives the request to start an experiment. The input data is the folder name where input ECG files are located. The experiment name is the folder name with input files.
2. Files names are read from the folder.
3. The file name from the list is sent to EcgAuth.EcgProcessingApi to start processing.
4. EcgAuth.EcgProcessingApi downloads the file from the bucket and processes it. Then downloads fake coordinates. From found ECGs coordinates and fake ones creates a file for machine learning.
5. Created file for machine learning is uploaded to the storage.
6. Response from EcgAuth.EcgProcessingApi informs EcgAuth.ExperimentRunnerApi that processing ECG file is complete. The response contains a GUID. It identifies the person in the system to whom ECG belongs.
7. The request to learn neural models is sent to EcgAuth.LearningRunnerApi. The body of the request contains the experiment name and the GUID of the person. This data is required to find the file with coordinates in the storage.
8. EcgAuth.LearningRunnerApi downloads the file with coordinates and makes 10 pairs of data to learn 10 neural models.
9. The request is sent to EcgAuth.MachineLearning.EngineApi to create one neural model. The body contains the experiment name, the GUID and objects that machine learning uses. The experiment name and the GUID are needed to upload a created neural model to the correct folder in the storage.
10. EcgAuth.MachineLearning.EngineApi uploads the created neural model to the storage.
11. Receiving the response that means learning a neural model is done.
12. Receiving the response that means the selected ECG is registered in the system.

5. Experiments

The web service was deployed to the university (IMMSP NASU) server in order to validate its workability and efficiency. For this need a virtual machine was launched there. Kubernetes was used to deploy instances of microservices to the virtual machine. Microservices were accessible through the internet.

Only one instance for each microservice has been deployed to the cloud. Horizontal scaling for microservices hasn't been configured, so all experiments have this fixed set of instances. Each microservice could use all computing cores allocated to the virtual machine. Experiments were made with 2 and more allocated cores to the virtual machine.

Knowing the fact that neural models calculations take 77% of all time, the learning of neural models needs to be parallelized. EcgAuth.LearningRunnerApi can control the number of models that learn simultaneously with the parameter. This parameter was specified for each experiment. All other requests among microservices were sequential.

Important to mention that the machine learning library ML.NET has a parallelization mechanism. It performs calculations concurrently in case it has access to more than 1 core. That means that ML.NET will use all allocated cores to the virtual machine when learning one neural model.

All experiments used the same set of 30 ECGs as input data. Total execution time is used as an indicator to measure efficiency. Total time is the time required to register all ECGs from the input dataset. One more important indicator is the average time to learn one neural model. This indicator shows how learning models simultaneously makes an effect on machine learning computing. Total execution time and average time to learn one neural model are used to make conclusions about architecture efficiency.

The first experiment was executed with the following configurations. All requests were sequential which means all neural models were learned sequentially as well. The virtual machine had access to 2 server cores. The total time was 12 min 8 sec and the average time to learn one model was 2.1 sec. This result is the main one that was used to compare with others since it didn’t use any parallelization to improve efficiency, only the ML.NET library did some parallel computations under the hood.

The next experiments had changed the parameter that is responsible for the amount of concurrently trained models. In the second experiment 2 models learned concurrently, in the third 3 models and so on. The result is presented in Table 1.

Table 1

The experiments results with two allocated cores to the virtual machine

Allocated cores	Simultaneously learned models	Total experiment time for 30 ECGs	Average time to learn one model
2	1	12 min 8 sec	2.1 sec
2	2	10 min 31 sec	3.2 sec
2	3	9 min 2 sec	4 sec
2	4	8 min 59 sec	5.2 sec
2	5	8 min 47 sec	6.4 sec

As we can see by adding more concurrency the total experiment time dropped down with every experiment and the average time to learn one model went up a little. Presented only 5 experiments since the 6th experiment had the result close to the 4th and the 5th. The total time dropped down since learning neural models was becoming more concurrent with each experiment. The average time to learn one model went up since concurrently training neural models can’t learn all time on the limited computation resources, they need to wait for each other from time to time.

The next step was to allocate 2 more cores to the virtual machine. Additional computation resources had to improve total execution time. For the following experiments 4, 6, and 8 cores were allocated to the virtual machine. All experiment results are presented in Table 2.

Table 2

All experiments results

Allocated cores	Simultaneously learned models	Total experiment time for 30 ECGs	Average time to learn one model
2	1	12 min 8 sec	2.1 sec
2	2	10 min 31 sec	3.2 sec
2	3	9 min 2 sec	4 sec
2	4	8 min 59 sec	5.2 sec
2	5	8 min 47 sec	6.4 sec
4	1	12 min 31 sec	2.2 sec
4	2	8 min 15 sec	2.7 sec
4	3	7 min 51 sec	3.5 sec
4	4	7 min 19 sec	4.1 sec
4	5	6 min 59 sec	5 sec
6	1	15 min 28 sec	2.8 sec
6	2	9 min 39 sec	3.2 sec
6	3	8 min 52 sec	3.9 sec
6	4	8 min 48 sec	4.7 sec
8	1	20 min 3 sec	3.7 sec
8	2	12 min 18 sec	4.2 sec
8	3	10 min 29 sec	4.7 sec
8	4	9 min 33 sec	5.8 sec

The results show that increasing computation resources improve total execution time. With allocated 4 cores and concurrently trained 5 neural models the total registration time of all ECGs became 6 min 59 sec. By manipulating the number of cores and the number of concurrent trained models it was possible to reduce total time for 5 min 9 sec which is 42%.

We can see that the more cores were allocated to the virtual machine the worse the total execution time became. It's easy to see by analyzing experiments where only one model learned concurrently. The results show that the more the ML.NET library had access to the cores the slower it trained one model.

This issue with parallel computation is well known. There is Amdahl's law [29]. It says the execution time can be improved using concurrent calculations but it can't be better after some level of parallelization. It happens in this way since concurrent execution consists of sequential work, so it can't be less than it. If concurrent calculations are supported by the application then allocating more cores should improve the performance. Analyzing the experiments where concurrently trained 4 models we can see the improvement after adding 2 more cores. Comparing experiments with 2 and 4 cores the time was reduced for 1 min 40 sec (19%). But for this architecture providing more computing power didn't make productivity better or kept staying on some level. Using 4 cores is the configuration setup when the best performance has been reached.

Amdahl's law says the reason why performance can drop with parallel executions. Any parallel executions are orchestrated by sequential work. The ML.NET library uses all of the accessible cores of the microservice instance, which means a lot of parallel work happens under the hood. This slowed down the process. It's easy to see on experiment results where concurrently trained only 1 neural model. The average time to learn one model increased for 1.6 sec or 76% between 2 and 8 used cores. The more cores have ML.NET, the more sequential work happens to coordinate parallel executions, instead of learning models the time spent to organize calculations.

For the setup where 1 instance was deployed for each microservice, the total execution time was improved with the help of parallelization. Due to the specific work of the ML.NET library, additional allocated resources started to make delays that after experiments with 4 cores leveled the profit of concurrent calculations. Timing results confirmed Amdahl's law.

6. Future researches

Before the experiments, there was the expectation that by adding more computing power the experiment time would drop down to some moment. Because of the way ML.NET works, with adding more cores the performance was dropping down. The obtained time improvements can be better. Changes need to be done to the web service to improve performance. One of the ways is to try another machine learning library but a better idea is to change the microservice architecture setup by adding new instances of microservices.

The most time-consuming part after improvements is still machine learning. The next step is to deploy additional instances of EcgAuth.MachineLearning.EngineApi. Knowing the way ML.NET works, it is better to deploy 10 instances of that microservice and provide access only to one core of the virtual machine. In this case, ML.NET won't spend time coordinating between cores.

The next step is improving web service by adding authentication and a mechanism to evaluate authentication results. These functionalities also would be provided by microservice and deployed to the cloud. It will help to evaluate the efficiency of authentication technology and provide a comfortable tool to work with. It will make research more organized and help improve the web service iteratively.

7. Conclusions

The part of authentication technology (registering ECGs) was presented as a web service and deployed into a cloud. The developed architecture can work with a big set of data. The way architecture is built allows improving productivity by adding parallelization. To evaluate efficiency the total time to register 30 ECGs was used. The execution time without parallelization was 12 min 8 sec and the best experiment time with parallelization was 6 min 59 sec. The total experiment time decreased by 42% which is a good result.

The architecture allows better performance by having more computing resources but the total execution time and the average time to learn one neural model show decreasing in performance with allocating more cores to the virtual machine. For the current setup where each microservice has 1 instance, efficiency started to drop down after allocating 6 cores to the virtual machine. The reason for that is the ML.NET library. It uses all accessible cores that create delays since the library switches threads between cores while calculating. The experiment results confirmed Amdahl's law which says that a program's performance can be improved to some moment by adding concurrency. The described issue can be removed by deploying new instances to learn neural models and limiting them to use only one core.

Having registration of ECGs as a web service is good for research since it provides more computing resources. All researchers work with one product version which simplifies and organizes work.

The next works involve using a different deploy setup for experiments and adding authentication with its evaluation.

8. References

[1] Mohit Ingale, Renato Cordeiro, Siddartha Thentu, Younghee Park, Nima Karimian. "ECG Biometric Authentication: A Comparative Analysis." IEEE Access, (2020): 117853-117866. doi: 10.1109/ACCESS.2020.3004464

[2] Nikita Samarin, Donald Sannella. A Key to Your Heart: Biometric Authentication Based on ECG Signals, Hyatt Regency Santa Clara, Santa Clara, United States 11 Aug 2019

[3] Jin Su Kim, Sung Hyuck Kim, and Sung Bum Pan. Personal recognition using convolutional neural network with ecg coupling image. Journal of Ambient Intelligence and Humanized Computing 11 (2020) 1923–1932,. doi: 10.1007/s12652-019-01401-3

[4] Majid Komeili, Wael Louis, Narges Armanfard, and Dimitrios Hatzinakos. "Feature selection for nonstationary data: Application to human recognition using medical biometrics." IEEE transactions on cybernetics 48.5 (2017): 1446-1459. doi: 10.1109/TCYB.2017.2702059

[5] Konstantinos N Plataniotis, Dimitrios Hatzinakos, and Jimmy KM Lee. “Ecg biometric recognition without fiducial detection.” In 2006 Biometrics symposium: Special session on research at the biometric consortium conference (2006). doi: 10.1109/BCC.2006.4341628.

[6] Somsanouk Pathoumvanh, Surapan Airphaiboon, and Kazuhiko Hamamoto. Robustness study of ecg biometric identification in heart rate variability conditions. IEEJ Transactions on Electrical and Electronic Engineering 9.3 (2014) 294–301. doi: 10.1002/tee.21970.

[7] Qingxue Zhang, Dian Zhou, and Xuan Zeng. “Heartid: A multiresolution convolutional neural network for ecg-based biometric human identification in smart health applications.” Ieee Access 5 (2017): 11805-11816. doi: 10.1109/ACCESS.2017.2707460

[8] Robin Tan and Marek Perkowski. Toward improving electrocardiogram (ecg) biometric verification using mobile sensors: A two-stage classifier approach. Sensors 17.2 (2017): 410. doi: 10.3390/s17020410

[9] Pei-Lun Hong, Jyun-Ya Hsiao, Chi-Hsun Chung, Yao-Min Feng, and Shun-Chi Wu. “Ecg biometric recognition: template-free approaches based on deep learning.” In 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (2019): 2633-2636. doi: 10.1109/EMBC.2019.8856916

[10] Ruggero Donida Labati, Enrique Muñoz, Vincenzo Piuri, Roberto Sassi, and Fabio Scotti. Deep-ecg: convolutional neural networks for ecg biometric recognition. Pattern Recognition Letters 126 (2019) 78–85. doi: 10.1016/j.patrec.2018.03.028

[11] João Ribeiro Pinto, Jaime S Cardoso, André Lourenço, and Carlos Carreiras. Towards a continuous biometric system based on ecg signals acquired on the steering wheel. Sensors 17.10 (2017) 2228. doi: 10.3390/s17102228

[12] Yazhao Li, Yanwei Pang, Kongqiao Wang, and Xuelong Li. Toward improving ecg biometric identification using cascaded convolutional neural networks. Neurocomputing 391 (2020) 83–95. doi: 10.1016/j.neucom.2020.01.019

[13] Vu Mai, Ibrahim Khalil, and Christopher Meli. “Ecg biometric using multilayer perceptron and radial basis function neural networks.” In 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2011): 2745-2748. doi: 10.1109/IEMBS.2011.6090752

[14] Song-Kyoo Kim, Chan Yeob Yeun, and Paul D Yoo. “An enhanced machine learning-based biometric authentication system using rr-interval framed electrocardiograms.” IEEE Access 7 (2019): 168669-168674. doi: 10.1109/ACCESS.2019.2954576.

[15] Nima Karimian, Zimu Guo, Mark Tehranipoor, and Domenic Forte. “Highly reliable key generation from electrocardiogram (ecg).” IEEE Transactions on Biomedical Engineering, 64.6 (2016):1400-1411. doi: 10.1109/TBME.2016.2607020

[16] Kuikui Wang, Gongping Yang, Yuwen Huang, and Yilong Yin. Multi-scale differential feature for ecg biometrics with collective matrix factorization. Pattern Recognition 102.8 (2020) 107211. doi: 10.1016/j.patcog.2020.107211

[17] Sairul I Safie, John J Soraghan, and Lykourgos Petropoulakis. “Electrocardiogram (ecg) biometric authentication using pulse active ratio (par).” IEEE Transactions on Information Forensics and Security, 6.4 (2011):1315-1322. doi: 10.1109/TIFS.2011.2162408

[18] Jun Shen, Shu-Di Bao, Li-Cai Yang, and Ye Li. “The plr-dtw method for ecg based biometric identification.” In 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, (2011): 5248-5251. doi: 10.1109/IEMBS.2011.6091298

[19] Nima Karimian, Mark Tehranipoor, Damon Woodard, and Domenic Forte. “Unlock your heart: Next generation biometric in resource-constrained healthcare systems and iot.” IEEE Access, 7 (2019): 49135-49149. doi: 10.1109/ACCESS.2019.2910753

[20] Emna Kalai Zaghouani, Adel Benzina, and Rabah Attia. “Ecg based authentication for e-healthcare systems: Towards a secured ecg features transmission.” In 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), (2017): 1777-1783. doi: 10.1109/IWCMC.2017.7986553

[21] V. Vishnevsky, T. Romanenko, L. Kizub, Biometric identification of a person according to his/her electrocardiogram, Mathematical Machines and Systems 2 (2018) 88-95.

[22] V. Vishnevsky, V. Kalmykov, T. Romanenko, Approximation of one-, two- and three-dimensional curve arcs by parametric splines, Mathematical Machines and Systems 4 (2015) 57-64.

[23] V. Vishnevsky, T. Romanenko, Application of the hausdorf metrics for determining atypical cardiocycles in three-dimensional vectorcardiogram phase space coordinate, Medical Informatics and Engineering 3 (2019) 31-36.

[24] Vishnevsky V., Romanenko T., Kizub L. Experimental verification of possibility of human identification by the electrocardi-ogram. 5th International Conference on Application of Information and Communication Technology and Statistics and Economy and Education *(ICAICTSEE – 2015)* (2015) 318.

[25] V.V. Vyshnevskyi, T.M. Romanenko, L.A. Kizub, The use of cardiograms and their characteristics for human identification. Visnyk of Vinnytsia Politechnical Institute 5 (2016) 7-10.

[26] V. Vishnevsky, T. Romanenko, Y. Luhovskyi, Validity of human authentication by electrocardiogram with a limited number of channels, Mathematical Machines and Systems 2 (2015) 57-64.

[27] V. Vishnevsky, A method of automatic authentication of a person based on their electrocardiogram Patent, 2018. No. 117713, Filed Feb. 15th, 2017, Issued Sep. 10th., 2018.

[28] V. Vishnevsky, T. Romanenko, Y. Luhovskyi, O. Boretsky, Data preparation services for the authentication of people by their electrocardiogram. Mathematical machines and systems 3 (2022) 58–69. doi: 10.34121/1028-9763-2022-3-58-69

[29] Gene Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the afips conference proceedings, vol. 30 (atlantic city, n.j., apr. 18–20).” Solid-State Circuits Newsletter, IEEE 12.3 (2007): 19-20. doi: 10.1109/N-SSC.2007.4785615