Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 2202–2211

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Deep learning for heart disease detection through cardiac sounds

Luca Brunese[a], Fabio Martinelli[b], Francesco Mercaldo[a,b,*], Antonella Santone[c]

[a] *Department of Medicine and Health Sciences "Vincenzo Tiberio", University of Molise, Campobasso, Italy*
[b] *Institute for Informatics and Telematics, National Research Council of Italy (CNR), Pisa, Italy*
[c] *Department of Biosciences and Territory, University of Molise, Pesche (IS), Italy*

Abstract

Most of death causes are related to cardiovascular disease. In fact, there are several anomalies afflicting the heart beat, for instance heart murmur or artefact. We propose a method for heart disease detection. By gathering a set of feature obtainable directly from cardiac sounds, we consider this feature vector as input for a deep neural network to discriminate whether a cardiac sound is belonging to an healthy or to a patient with a cardiac disease. The experiment we performed demonstrated the effectiveness of the proposed approach in real-world environment.

Keywords: heart disease, heart beat, deep learning, classification

1. Introduction and Related Work

According to the Center for Disease Control and Prevention every 37 seconds one person loses his/her life due to cardiovascular diseases [22]. Lifestyle can conduct to heart disease, for instances obesity and overweight, unhealthy diet and excessive use of alcohol.

The United Kingdom National Health Service considers several test to diagnose heart-related problems, for instance electrocardiogram, exercise stress tests, coronary angiography or radionuclide tests. All these techniques require people to physically go to hospital facilities.

In this paper we propose a methodology to detect cardiac disease by analysing cardiac sounds. Cardiac sounds can be obtained by doctors, using a digital stethoscope but also directly from the patients, by exploiting his/her smart devices (for instance a tablet or a smartphone) equipped with an adequate mobile application. Cardiac sounds are converted in numeric values (features) and thus are used as input for a deep learning classifier to detect whether the cardiac sound is belonging to an healthy patient or to a patient with a cardiac disease.

The aim of this paper is to provide a method for a first level of screening related to cardiac pathologies.

* Francesco Mercaldo
E-mail address: francesco.mercaldo@unimol.it

Several works in current literature are related to the automatic detection of cardiac disease [11]. For instance, authors in [20] propose a predictive model for heart disease diagnosis using a fuzzy rule-based approach with decision tree. Researchers in [1] evaluated six different machine learning algorithms to predict the presence of coronary artery disease. Neural networks are considered by Gu et al. [15], regression analysis applications for myocardial infarct localisation are considered in [10]. Authors in [17] grading myocardial motion defects, while researchers in [16] are focused in estimating disease from heart volumes.

The main difference with the cited work and the approach we propose is represented by the user of the propose method: in fact, our approach is the first one providing a first level of screening obtainable directly from the patient. As a matter of fact, the cardiac sounds are easily obtainable by exploiting a mobile device.

The paper proceeds as follows: next section describes the proposed method, Section 3 presents the results of the experimental analysis on a real-world dataset and, finally, conclusion and future plan of research are discussed in Section 4.

2. The Method

In this section we describe the method we propose for the automatic detection of hearth diseases starting from cardiac sounds. We firstly describe the feature vector we propose and thus we discuss the deep learning network we designed.

2.1. The Feature Vector

Table 1 describes the feature vector involved in the following study.

Table 1: The Features.

#	Feature	Description
F_1	*chroma_stft*	Compute a chromagram from a waveform
F_2	*spectral_centroid*	Mean of the magnitude spectrogram
F_3	*spectral_bandwidth*	Compute the spectral bandwidth
F_4	*zero_crossing_rate*	Compute the zero-crossing rate
F_5	*mfcc*	Mel-Frequency Cepstral Coefficients

In particular, F_5 represents a groups of features. In fact, the Mel-Frequency Cepstral Coefficients (i.e., F_5) represents one of the most widespread method to extract numerical features from an audio signal and is used majorly whenever working on audio signals. The mel frequency cepstral coefficients of a signal are a small set of features (usually 20) which concisely describe the overall shape of a spectral envelope. In this work the F_5 feature groups 20 mel frequency cepstral coefficients (indicated with $mfcci$ with 1 ≤ i ≤ 20).

2.2. The Deep Learning Model

Figure 1 shows the deep learning architecture we designed. We adopt a sequential model i.e., a linear stack of layers.

The architecture is composed mainly by two kind of layers: one *InputLayer* and four *DenseLayer*. An *InputLayer* is aimed to send the feature vectors into the network. Subsequently, considering that the input data are represented as vectors we consider *densely connected layers* i.e., *DenseLayer* with a *relu* activation function.

To understand what is happening into the deep network when is composed by a stack of several *DenseLayer*, let us consider the following code snippet for the implementation of a *DenseLayer*:

```
layer = layers.Dense(32, input_shape=(784,))
```

The instruction represents the definition of a *DenseLayer* accepting a 784 sized vector and will return a 32 sized vector (a dense layer with 32 output units). As a consequence, the next layers will be declared in such a way that it is able to accept as input a 32 sized vector so building the layer chain.

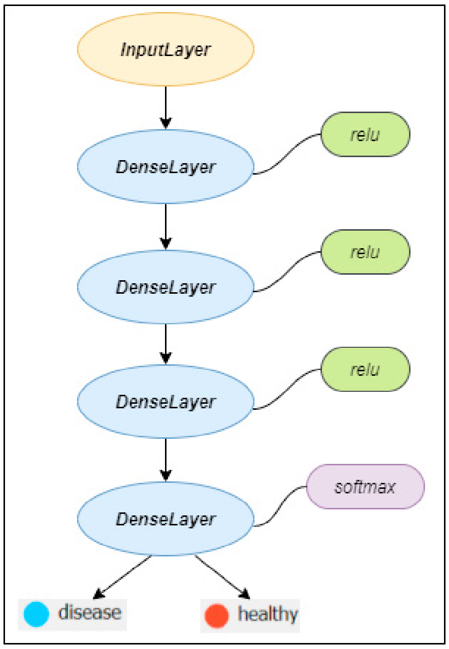

Fig. 1: The deep learning architecture. In yellow the *InputLayer* , in blue the *DenseLayer*, in green the activation function we set for three *Dense-Layer* (i.e.,*relu*) and in purple the activation function for the last *DenseLayer* (i.e.,*softmax*).

The purpose of the layer is to transform the data. In the following we explain how these transformations can take place: a more complete declaration of the *DenseLayer*, previously considered, is the following one:

layer = layers.Dense(16, input_shape=(784,), activation='relu')

The argument passed to *DenseLayer* is 16 in this case and it represents the number of hidden units of layers (an output 16 sized vector): formally a hidden unit is a dimension in the representation space of the layer.

In this layer the function able to "resize" the data has been specified (i.e., activation='relu' and it is called *activation function*). The layer can be interpreted as follows:

output = relu(dot(W, input)+b)

The *output* value is the combination of several operations: a dot product (i.e., *dot*) between the input vector and W (the weights of the layer) and an addition between the resulting vector and the bias (i.e., *b*). While the relu and the addition are element-wise operations, the dot operation combines entries (this is the operation that consents to the layer to "resize" the output data) [13]. Considering that W represents the actual value of weights for the considered layers (considering that the number of the weight is equal to the number of the neurons, for this reason whether we want to transform the data into a 16 size vector, the layer exhibits 16 weight, one weight for each neuron), while input represents the input feature vector.

The first three *DenseLayer* of the proposed neural network consider *relu* as activation function.

We consider the *relu* activation function because it allows the network to converge very quickly with the backpropagation.

In addition the relu function, frequently considered as candidate for intermediate layers, another widespread activation function, the so-called softmax, is usually considered as last layer in classifiers neural network based before the output layer.

For instance, whether an instance under analysis can belong to one of the 2 label considered (the label in the proposed method are *disease* and *healthy*), it is possible to add following layer as final one:

layer = layers.Dense(2, activation='softmax')

The input from the previous layer is resized in a 2 size vector, the only difference is that the resizing is done using the softmax function and not the relu one.

The reason why the softmax activation function is considered as last layer before the output one is that its aim is to normalize an input vector into a probability distribution: it is usual that vector can exhibit negative or greater than one values. In order to have a feature vector which elements summed are equal to one we apply the softmax function. For instance, by considering as example a case with four labels, whether the output of the previous *DenseLayer* with the softmax activation function is equal to $[0.2, 0.1, 0.4, 0.3]$ this is symptomatic that the instance under analysis is belonging to the #3 class (the first element of the output vector presents the probability that the instance is belonging to the first class and so on).

The last *DenseLayer* in the proposed deep learning architecture considers a *softmax* activation function. We consider the *softmax* activation function because is able to handle multiple classes in only one class, and divides by their sum, giving the probability of the input value being in a specific class. It is typically used only for the output layer, for neural networks that need to classify inputs into multiple categories.

2.3. Study Design

The evaluation consists of three stages: (i) a comparison of descriptive statistics of the populations of patients; (ii) a comparison of distribution of the considered features ; and (iii) a classification analysis aimed at assessing whether the exploited features are able to correctly discriminate between healthy and patient afflicted by cardiac disease. The (i) and (ii) tasks of the study design was accomplished with Orange framework, a suite of machine learning software [9], largely employed in data mining for scientific research. With regard to the (iii) task, the model was developed by exploiting Keras[1], a high-level neural networks API, written in Python and capable of running on top of TensorFlow[2], a machine learning open-source software library for both research and production at Google.

3. Experimental Analysis

In following section the results of the experiment, aimed to demonstrated the effectiveness of cardiac sounds from discriminating between healthy and patients heart disease affected, we performed are presented.

To evaluate the proposed method, we consider a dataset composed by real cardiac sounds obtained by the *Classifying Heart Sounds Challenge*[3]. The heartbeat sounds are gathered by exploiting the iStethoscope Pro iPhone app[4]. The app exploits the audio capabilities of modern mobile devices, performing realtime filtering and amplification, and enabling users to view Fast Fourier transform (FFT) spectrograms and email eight seconds of audio. The quality of the audio as assessed by the cardiologists is as good as or better than commercially available digital stethoscopes [12]. The dataset contains heartbeats related to following disease categories: murmur (heart sounds produced when blood is pumped across a heart valve), extrasystoles (additional heartbeats that occur outside the physiological heart rhythm) and artifact (disturbances in rhythm monitoring). 145 heartbeats disease related are considered and 31 heartbeats recorded by *healthy* patients, for a total of 176 heartbeats considered in the experimental evaluation.

The machine used to run the experiments and to take measurements was an Intel Core i7 8th gen, equipped with 2GPU and 16Gb of RAM.

The remaining of the section reflects the stages described in the study design.

3.1. Descriptive Statistics

For the descriptive statistics comparison we consider scatterplots i.e., a plot considered for showing the relationship between two numerical features. We consider scatterplot with the aim to provide a graphical impact about the numerical values obtained from different features under analysis: as a matter of fact, the lower the overlapping of the

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] http://www.peterjbentley.com/heartchallenge/
[4] https://apps.apple.com/it/app/istethoscope-pro/id322110006

numerical values assumed by different numerical features, the greater the probability of obtaining a classifier with good prediction accuracy. For space reason, few cases are represented and discussed, but similar consideration can be made for all the numeric features involved in the study.

Figure 2 shows the scatterplot for *mfcc20* and *mfcc14* features: the red points are related to *healthy* patients, while the *blue* one to patients with a cardiac *disease*.

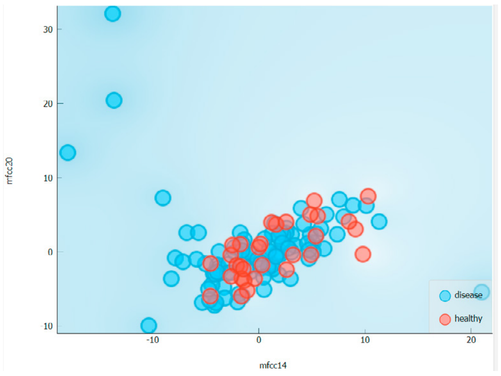

Fig. 2: Scatterplot for *mfcc20* and *mfcc14* features.

From the scatterplot in Figure 2 it seems that the *disease* instances ranges in a wider spaces if compared to the *healthy* one. This can be symptomatic that healthy persons produce similar cardiac sounds if compared to persons afflicted by a cardiac *disease*.

Figure 3 shows the scatterplot for *mfcc7* and *mfcc19* features.

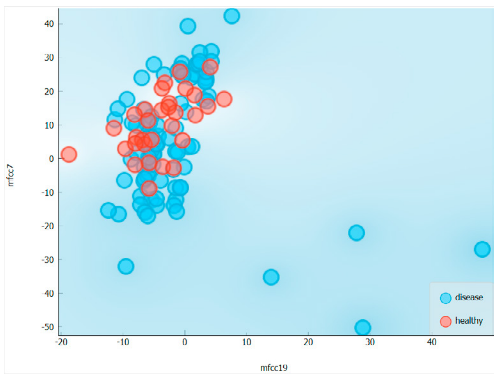

Fig. 3: Scatterplot for *mfcc7* and *mfcc19* features.

Also in this second scatterplot two Mel-frequency cepstral coefficients are considered (i.e., *mfcc7* and *mfcc19*). Coherently with the scatterplot in Figure 2, both the instances related to *healthy* and patient affected by a cardiac *disease* are falling in a similar area in the scatterplot. We note that the *healthy* instances exhibit similar numeric values if compared to the *disease* instances.

In Figure 4 is shown the scatterplot for the *mfcc6* and the F_4 features.

Even in this case the majority of the *healthy* and *disease* instances are localised in the same area i.e., in the left side of the scatterplot. In particular the *healthy* instances present numeric value in a similar range if compared with the *disease* ones.

The last scatterplot we present is the one depicted in Figure 5: it represents *mfcc9* and F_1 features.

Coherently with scatterplots in Figures 2, 3 and 4 also in this scatterplot the *healthy* and the *disease* instances are grouped in the same area, with a concentration in similar numeric values for the *healthy* instances.

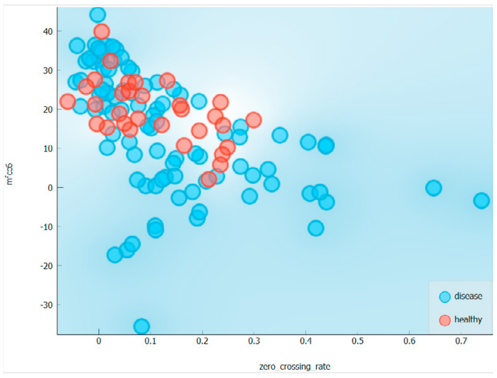

Fig. 4: Scatterplot for *mfcc6* and F_4 features.

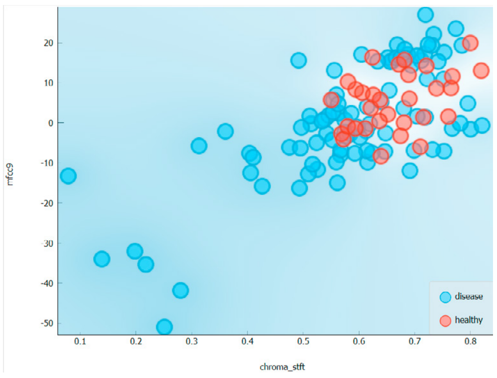

Fig. 5: Scatterplot for *mfcc9* and F_1 features.

3.2. Distribution Comparison

In the following we analyse the numeric features distribution, with the aim to highlight whether there are differences from the *healthy* and *disease* afflicted patients from the point of view of the numeric features we consider. The idea behind the distribution analysis is to understand whether the single features exhibits similar values between the *healthy* and *disease* patients distributions.

In Figure 6 is shown the distribution related to *healthy* and *disease* affected patients for the F_2 feature.

As evidenced from the scatterplot in Figure 6 the distribution of the *healthy* patients assumes a more restricted set of values if compared to the ones obtained from the *disease* afflicted patients.

The F_3 feature scatterplot is depicted in Figure 7.

Also this feature follows a trend really closer to the one exhibited from the F_2 feature in the scatterplot in Figure 6.

Figure 8 shows the distribution related to the $mfcc1$ feature.

For *healthy* patients the $mfcc1$ feature is approximately ranging between -800 and -100, while for *disease* affected patients the same feature is ranging between -1200 and -370. This is confirming that *disease* instances span in wider interval with respect to *healthy* instances.

In Figure 9 the distribution for the $mfcc10$ feature is shown.

Coherently with the distributions of the other features, also the *healthy* instances for the $mfcc10$ features range in a smaller intervals if compared to the ones of the *disease* affected patients.

3.3. Classification Analysis

For model building and evaluation, we split the dataset into training and testing part in 80:20 ratio i.e., a hold-out validation is considered.

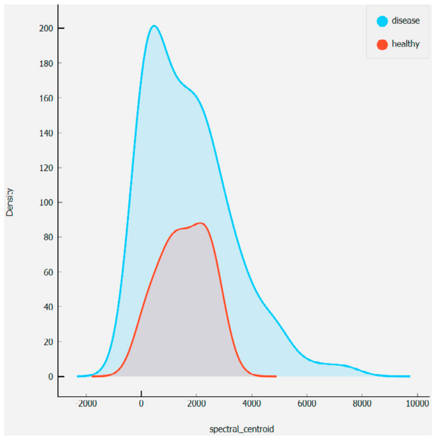

Fig. 6: Distribution for the F_2 feature.

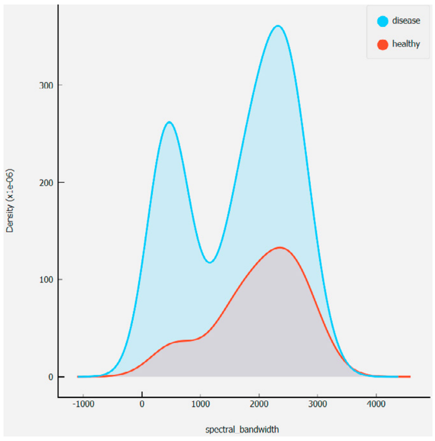

Fig. 7: Distribution for the F_3 feature.

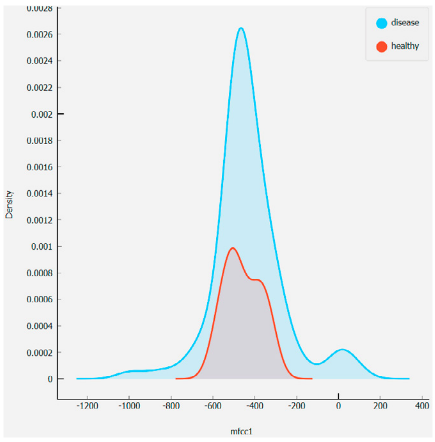

Fig. 8: Distribution for the $mfcc1$ feature.

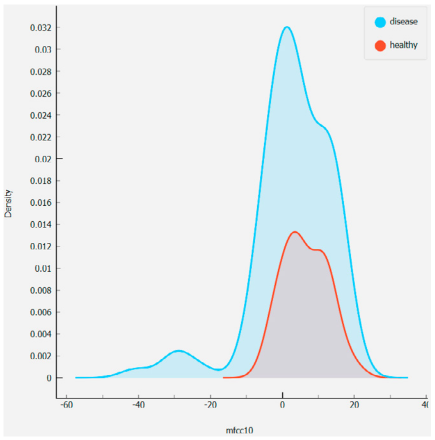

Fig. 9: Distribution for the $mfcc10$ feature.

We consider two metrics to evaluate the performances of the classification: accuracy and loss.

The accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's true value: it is the fraction of the classifications that are correct and it is computed as the sum of true positives and negatives divided all the evaluated instances:

$$Accuracy = \frac{tp+tn}{tp+fn+fp+tn}$$

where *tp* indicates the number of true positives, *tn* indicates the number of true negatives, *fn* indicates the number of false negatives, *fp* indicates the number of false positives.

The Loss is a quantitative measure of how much the predictions differ from the actual output (i.e., the label). Loss is inversely proportional to the correctness or the model.

The loss is calculated on training and validation and its interpretation is how well the model is doing for these two sets: it is a summation of the errors made for each example in training or validation sets.

From the Accuracy and Loss definitions, it is expected that Accuracy and Loss should be inversely proportional: for high values of accuracy, low loss values are expected (and the opposite). Furthermore, considering that the weights and bias are initially random selected, the accuracy trend should start by exhibiting low values (and high loss value, symptomatic that the network is performing wrong predictions), but whether the network during the several "epochs" (i.e., one forward pass and one backward pass of all the training examples) is able to learn (i.e., it is able to solve the driver prediction problem), the accuracy should start to exhibit higher values in the next iterations (and consequently the loss should exhibit low values). The epoch is a parameter chosen by the network designer, usually the number of epochs chosen is such that the loss is at least and it does not get worse in the immediately succeeding epochs and, consequently, the accuracy value reached is the maximum and in the immediately succeeding epochs is not improving, symptomatic that the network has reached the stability and that further epochs would not improve performances. We set the number of epochs equal to 100, because the network reached the stability with a number less or equal to 100.

Figure 10 show the accuracy values when the number of epoch is increasing.

After 87 epochs the network reaches the stability with an accuracy equal to 0.9899. The research paper in [12] consider the same dataset with supervised machine learning techniques [18, 8] obtaining lower performances if compared to the proposed method: as a matter of fact they obtain a precision equal to 0.71 with the J48 algorithm, to 0.92 with the MLP algorithm and to 0.58 with the UCL algorithm, confirming the effectiveness of deep learning for improving the performances.

Figure 11 shows the loss values when the epoch number is increasing.

The loss continues to decrease when the epoch number is increasing, it reaches a value equal to 0.788 for the 100-th epoch.

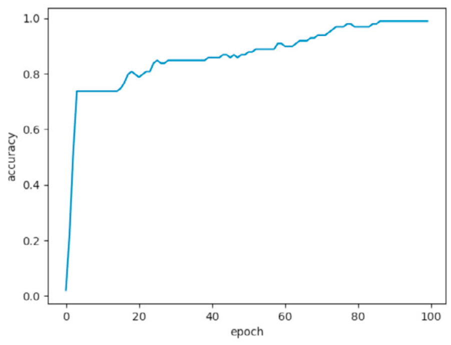

Fig. 10: Accuracy plot.

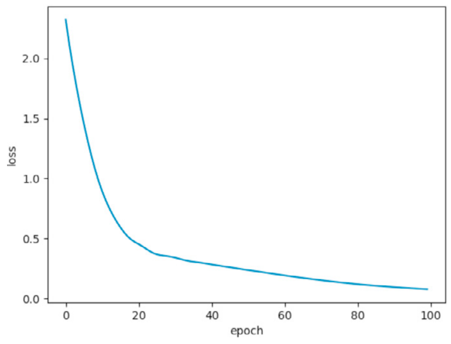

Fig. 11: Loss plot.

4. Conclusion and Future Work

With the aim to provide a method for the first screening level of cardiac pathologies, in this paper we propose an approach to determine directly from heartbeat sounds whether a patient is afflicted by a heart *disease* or not. The heartbeat sounds can be easily obtained from the patient, in fact the dataset evaluated in the study is obtained with a smartphone equipped with the iStethoscope Pro mobile app. An accuracy of 0.98 is reached, demonstrating the effectiveness of the proposed method to discriminate between *healthy* and *disease* affected patients. As future work we plan to evaluate the proposed method on the detection of specific cardiac pathologies. Moreover, we will investigate whether the proposed methods can obtain good performance also on other type of sounds obtainable from digital stethoscopes for instance, sounds gathered from lung [3]. We will also investigate if formal verification techniques [21, 14] can obtain better performances as demonstrated in similar contexts, from cancer detection [5, 2, 4] to malware detection [19, 7, 6].

ACKNOWLEDGEMENTS

This work has been partially supported by MIUR - SecureOpenNets and EU SPARTA contract 830892, Cyber-SANE projects, and the EU project CyberSure 734815.

References

[1] Akella, A.B., Kaushik, V., 2020. Machine learning algorithms for predicting coronary artery disease: Efforts toward an open source solution. BioRxiv .

[2] Brunese, L., Mercaldo, F., Reginelli, A., Santone, A., 2019a. Formal methods for prostate cancer gleason score and treatment prediction using radiomic biomarkers. Magnetic resonance imaging .

[3] Brunese, L., Mercaldo, F., Reginelli, A., Santone, A., 2019b. Neural networks for lung cancer detection through radiomic features, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–10.

[4] Brunese, L., Mercaldo, F., Reginelli, A., Santone, A., 2019c. Prostate gleason score detection and cancer treatment through real-time formal verification. IEEE Access 7, 186236–186246.

[5] Brunese, L., Mercaldo, F., Reginelli, A., Santone, A., 2020. An ensemble learning approach for brain cancer detection exploiting radiomic features. Computer methods and programs in biomedicine 185, 105134.

[6] Cimino, M.G., De Francesco, N., Mercaldo, F., Santone, A., Vaglini, G., 2020. Model checking for malicious family detection and phylogenetic analysis in mobile environment. Computers & Security 90, 101691.

[7] Cimitile, A., Martinelli, F., Mercaldo, F., 2017. Machine learning meets ios malware: Identifying malicious applications on apple environment., in: ICISSP, pp. 487–492.

[8] Cuzzocrea, A., Martinelli, F., Mercaldo, F., Vercelli, G., 2017. Tor traffic analysis and detection via machine learning techniques, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE. pp. 4474–4480.

[9] Demšar, J., Curk, T., Erjavec, A., Črt Gorup, Hočevar, T., Milutinovič, M., Možina, M., Polajnar, M., Toplak, M., Starič, A., Štajdohar, M., Umek, L., Žagar, L., Žbontar, J., Žitnik, M., Zupan, B., 2013. Orange: Data mining toolbox in python. Journal of Machine Learning Research 14, 2349–2353. URL: http://jmlr.org/papers/v14/demsar13a.html.

[10] Duchateau, N., De Craene, M., Allain, P., Saloux, E., Sermesant, M., 2016. Infarct localization from myocardial deformation: prediction and uncertainty quantification by regression from a low-dimensional space. IEEE transactions on medical imaging 35, 2340–2352.

[11] Duchateau, N., King, A.P., De Craene, M., 2020. Machine learning approaches for myocardial motion and deformation analysis. Frontiers in Cardiovascular Medicine 6, 190.

[12] Gomes, E.F., Bentley, P.J., Pereira, E., Coimbra, M.T., Deng, Y., 2013. Classifying heart sounds-approaches to the pascal challenge., in: HEALTHINF, pp. 337–340.

[13] Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep learning. volume 1. MIT press Cambridge.

[14] Gradara, S., Santone, A., Villani, M., 2005. Using heuristic search for finding deadlocks in concurrent systems. Information and Computation 202, 191–226. doi:10.1016/j.ic.2005.07.004.

[15] Gu, X., Jiang, Y., Ni, T., 2020. Discriminative neural network for coronary heart disease detection. Journal of Medical Imaging and Health Informatics 10, 463–468.

[16] Luo, G., Dong, S., Wang, K., Zuo, W., Cao, S., Zhang, H., 2017. Multi-views fusion cnn for left ventricular volumes estimation on cardiac mr images. IEEE Transactions on Biomedical Engineering 65, 1924–1934.

[17] Marin, T., Kalayeh, M.M., Pretorius, P.H., Wernick, M.N., Yang, Y., Brankov, J.G., 2011. Numerical observer for cardiac motion assessment using machine learning, in: Medical Imaging 2011: Image Perception, Observer Performance, and Technology Assessment, International Society for Optics and Photonics. p. 79660G.

[18] Medvet, E., Mercaldo, F., 2016. Exploring the usage of topic modeling for android malware static analysis, in: 2016 11th International Conference on Availability, Reliability and Security (ARES), IEEE. pp. 609–617.

[19] Mercaldo, F., Santone, A., 2020. Deep learning for image-based mobile malware detection. Journal of Computer Virology and Hacking Techniques , 1–15.

[20] Pathak, A.K., Valan, J.A., 2020. A predictive model for heart disease diagnosis using fuzzy logic and decision tree, in: Smart Computing Paradigms: New Progresses and Challenges. Springer, pp. 131–140.

[21] Santone, A., Vaglini, G., Villani, M., 2013. Incremental construction of systems: An efficient characterization of the lacking sub-system. Science of Computer Programming 78, 1346–1367.

[22] Ulbricht, T., Southgate, D., 1991. Coronary heart disease: seven dietary factors. The lancet 338, 985–992.