Multi-class 12-leads ECG beat classification: a recurrent approach combined with Convolutional Neural Network and Autoencoders

[redacted]

***Abstract*—Beat classification from electrocardiogram (ECG) is crucial for the prevention and diagnosis of cardiovascular diseases. Different deep learning approaches were developed in the recent years with the purpose to perform an automatic multi-class beat classification. To address this goal we propose two different models, one based on the Convolutional Neural Network and the other on Autoencoders approach. In order to take advantages from the temporal correlation of ECG signal we combine both of them with a RNN building two hybrid networks, CRNN (CNN + RNN) and ARNN (AE + RNN), that are compared on the base of classification performances and computational time. In this study, we demonstrate that both models produce good results and further, because of the different way to approach the signal, present different structures and organization. Moreover, most of the previous studies utilize only a subset of the all 12 leads given by ECG without any explanation. Thanks to ARNN, we tried to understand how the choice of a subgroup of leads influence the classification results.**

***Index Terms*—Unsupervised Learning, Recurrent Neural Networks, Deep Autoencoders, Convolutional Neural Networks, ECG Beat classification**

I. Introduction

Cardiovascular diseases (CVDs) are the leading cause of death worldwide: according to the World Health Organization (WHO), more people die annually from CVDs than from any other cause [1]; in fact, approximately 17.9 million people died from CVDs in 2016, representing 31% of all global deaths. CVDs are a group of heart disorders that can be divided in 3 main groups, concerning their different natures: electrical (due to dysfunctional electrical system of the heart), circulatory (blood vessels disorders, hypertension, aneurysm and so on) and structural (imputable to several diseases of heart muscle, such as cardiomyopathy). The primary method of monitoring, diagnosing and preventing such types of disorders is Electrocardiogram (ECG) that is the recording of heart electrical activity using electrodes placed on patient's limbs and chest, in specific and standard locations; thanks to this, ECG allows to measure the potential difference between two electrodes positions and generates the such called "leads"; the gold standard of this technique involves ten electrodes and generates an output with 12 leads, that are combined later in order to obtain a record where typical waveforms can be distinguished. In particular, in non pathological conditions, every heartbeat is composed by regular waveforms (P wave, QRS complex, T wave): based on the type of CVD, several irregularities can appear in one or more leads (such as abnormal intervals between waves or heartbeats, atypical morphological features, and so on). Typically, these abnormalities are identified by cardiologists mainly by visual inspection, involving time-wasting procedures that are complicated and also can't be error-prone. For this reason, in recent years, many researchers worked with the aim to develop computer-aided diagnosis (CAD) systems [2] and analysis methods in order to reduce diagnosis time, distinguish life-threatening from non-threatening events, improve cardiovascular diseases screening and also help patients in self-monitoring phases. Particularly, lot of progresses were achieved in this field thanks to Machine Learning techniques, mainly through Deep Learning tools. These one turns out to be very useful for heartbeat classification, made through a "beat labelling" procedure. The annotations that indicates the particular type of beat, referring to the standard Physionet set of codes [3]. Generally, the classification process of deep learning techniques is divided in three steps: i) data preprocessing, ii) features extraction (sometimes with feature selection) and finally iii) classification. As a matter of fact, features extraction is a fundamental part of heartbeat categorization that deeply influence the result of the whole process: "bad" features can lead to challenging classification that can also deteriorate system performances. In the field of Deep Learning, specifically, techniques such as Artificial Neural Networks (ANN) revealed to be very useful towards this types of problems, as they are able to extract automatically critical features (the so called "self-learning" ability) and have therefore satisfactory classification performances. Lot of researches demonstrated that automatic features extraction improves classifier efficiency, in particular in terms of speed and accuracy [4]. In this work, we developed two types of ANN, combining a Recurrent Neural Network (RNN) block with two different networks: a Convolutional Neural Network (CNN) in the first case, and Autoencoders (AE) in the second one; as we are going to notice later in this article, these networks differers from each other as a consequence of their different approach in features extraction and data manipulation. Specifically, it's widely demonstrated that RNN exhibit temporal dynamic behavior that allows them to capture temporal correlation in long sequences of input data, as they were sort of "memory cells"; because of the structure and behavior of ECG data, we considered this type of network to be suitable for our problem. Also, we develop

[redacted]

these two different types of networks in order to analyze and compare not only their classification performances, but also their computational costs: this parameter, in fact, tends to be critical in medical application, wearable devices and smart health. Moreover, very few articles in the literature make use of the information coming from all the 12 ECG leads, but tends to select two or few subgroups, probably legitimated by the lack of presence of studies on what are the most important or meaningful ones: based on this, we decided to use in our classification problem all the 12 ECG leads and then, eventually, study if there are some leads that are more relevant than others in order to obtain better classification results. This report is structured as follows. In section II we describe the state of the art; in section III, a high level description of our architectures is reported, followed by signals pre-processing techniques in section IV; in section V a more detailed explanation about the implemented networks is presented, with results explanation is section VI. The report is then closed with some final thoughts in the concluding remark section VII.

II. Related Work

Deep Learning techniques revealed to be in recent years resourceful and flexible tools to solve many different problems regarding an heterogenous mixture of fields. One can easily finds them applicated in computer vision and image classification [5], speech or handwriting recognition [6], but also in security or financial domain[7]; furthermore, like in our investigation, ANN are suitable for medical applications: ECG signal is de facto easy to acquire, but, due to its dimension and richness in features, its examination can be enhanced by computational analysis and instruments; following this reasoning, novel literature is experimenting a growing number of publications concerning ECG data. In the coming sections, we are going to illustrate, for the three networks we used, the main reports that mainly guided our research.

A. Convolutional Neural Network

Convolutional Neural Network is a widely used type of ANN. Acharya et al.[4] gives an example of how a simple CNN gives good performances in ECG beat classification. Authors underline how their CNN model is completely automated, so it does not require any hand-crafted feature extractor, selector or classificator. Moreover, their approach reveal to be able to analyze ”raw” ECG signals, meaning that it does not need any filtering or noise removal phase. In this study, however, it is considered only one ECG lead: in our opinion, using the information coming from all the 12 leads could bring an improvement in classifications results. Such this type of approach is applied in [8] where, a particular kind of CNN, called VGGNet, is used for detection and classification of arrhythmias (group of conditions where irregularities on heartbeat are present). More details and studies about this peculiar network can be found also in [9]; it consists mainly in a extremely deep CNN with a very huge number of layers. Despite the good classification performances, the authors doesn't take into account the computational cost of this huge number of convolutional layers: taking this reasoning as a starting point, in our work we develop a network that follows in principle the structure of the mentioned VGGNet, but without exceeding in the number of layers in order to obtain a smaller computational cost.

B. Deep Autoencoder

Autoencoders are a more recent approach than the conventionals CNNs and ANNs in general. They are exploited for severals aims, such as dimensionality reduction, data denoising or feature extraction. An Autoencoder is structured in principle in two parts: the ”encoder” part, where the network transform input data to lower dimensional representations (”coded” representation of the input); the ”decoder”, where the network, reconstruct a copy of the original input using the previous ”coded” representation of data. Thanks to their structure and rationale, in the literature one can find examples of AE application in speech enhancement for reduction of speech distortion [10], or image compression [11]; for what concern medical application, one can find usage of AE in lightweight lossy compression of biometric patterns [12], compression of ECG signals [13], ECG feature extraction [14], detection of cardiac arrhythmias [15], denoising [16] of medical images, and also in heartbeat classification [17],[18]; considering this last two works as a starting points, where the authors use autoencoders for feature extraction of ECG data coming from one ECG lead and then a deep neural network for heartbeat classification, we decided to implement an hybrid network with similar structure: we used in particular a series of AE working in parallel on all the 12 ECG leads for feature extraction, in order to verify if one can take advantages using information coming from the whole ECG signal or if there are some leads that contains more relevant features than others; this where followed also by a series of lead-specific RNN for beat classification.

C. Recurrent Neural Network

This type of ANN, thanks to its particular structure and algorithm, is used mainly when the input data are time series, where samples present strong correlation in time. RNN are mainly characterized by the presence of particular units, such as Long Short Term Memory layer (LSTM) or more recently Gated Recurrent Unit (GRU) units. Applications of this network regards language model implementations for speech prediction based on a given context [19], or again for electrocardiogram data classification [20]. In the present work, we decided to use the peculiar properties of GRU units on both the two networks we implemented.

Despite the huge number of studies using this particular Machine Learning techniques on ECG data for beat classification in particular, to the best of our knowledge we couldn't find studies that investigate about the difference using data coming from all the 12 leads or from a subset in terms of classification performances and computational burden. This reasoning was also one of the driven motivations of the present work.

III. Processing Pipeline

Our work focused mainly in investigating how beat classification performances could be enhanced by using informations coming from all the 12 ECG leads, in order to perform a discrimination process that takes into account the spatial and the temporal correlation between input data during the learning phase. Furthermore, we wanted to consider also the computational cost of this type of operation, in order to evaluate their use in medical devices with limited memory and power. We decided so to compare two types of architectures: a Convolutional-Recurrent Neural Network (CRNN) first, and on the other side a sort of hybrid network made by parallel Autoencoders that extract features specifically from each lead, followed by a Recurrent Neural Network that assign predicted labels to the different beat leads; then, thanks to a ”decision fusion” mechanism they are merged together to compute the final classification, as stated later in the article.

A. First network: CRNN Architecture

The architecture of our proposed model was made by four types of layers that performs specific functions on the whole input dataset that is entirely fed into the network: Convolutional, Max Pooling, Dropout and GRU layer (see Fig.[1]). Convolutional layers performs a one-dimensional (1D) convolution between the input data and a window of little size called ”kernel”; this window slide over the input data spatially performing a convolution on every input patch, acting as a sort of ”filter” that selects and transforms data in order to create feature maps that summarize the presence of those features in the input. Every convolution is followed also by several operations:

- Zero-padding: it is performed before the convolution, useful to maintains always the same input dimension;
- Batch normalization: corrects possible data distortions after the convolutional phase;
- Activation function: determinates the ”firing intensity” of the neuron in the network based on the result of the convolution of the data with kernel weights; this one in particular allows to assign to the input beats the predicted label;

Other layers present in the network are: the Dropout, a sort of regularization layer that avoid the so called overfitting phenomenon; the Max Pooling, that extract the maximum value of each patch of the input and generate an output with lower dimension. After the last convolutional layer, the features vector obtained is fed into the RNN, composed by a GRU unit and a Dense layer; this structure allows to take into account the temporal correlation of the input samples, unlike classic Feed Forward networks where indipendence between data is assumed. This type of approach revealed to be useful in when analyzing time series like ECG signal: the GRU layer, in fact, update its state vector using current input and the GRU state in the previous iteration; in this way the information concerning temporal data correlation flows during the whole computation process.

B. Second network: Lead-specific Autoencoders and Recurrent Neural Network (ARNN-12)

Implementing this second type of classification algorithm (from now on, referred as ARNN), we wanted to assess if beats classification can be enhanced by a ”lead-specific” feature extraction, taking advantage by the structure and functionalities of Autoencoders, and by the use of a series of recurrent neural networks (RNNs) as last layers. Our algorithm and its parts can be, at a high level, described as follows (full structure in fig.[2]). Discriminant features of each input beat were extracted by a series of Deep Autoencoders (DAEs), trained specifically on each lead: in this way we obtain 12 DAEs in parallel working on the different beats leads contemporary. Deep Autoencoders in general are a particular type of Autoencoders that, using several layers in a fully-connected structure, are able to ”code” the original input in an ”encoded” vector characterized by much smaller dimensions. This part, named ”encoder” is followed by a ”decoder” part, that map back these coded vector of features to an approximated reconstruction of the original input; thanks to the encoding phase, DAEs are able to memorize in the feature vector only useful properties of the data that can be used for classifications tasks. Each lead is compressed through a lead-specific trained encoder and passes from a dimension of (256,1) to a size of (32,1); the ”codes” are then fed, as features vectors, into 12 lead-specific RNNs that have the aim to classify each lead; RNNs are particular types of Feed Forward Neural Networks: here we implemented a very simple network made by a single GRU unit followed by a dense and an activation layer in order to obtain the predicted beat label. The final classification results are obtained then using a ”decision fusion” approach: final label of each beat was determined as the most probable class based on the assignments made by the different RNNs on the single leads. Taking advantage from this network architecture, we obtained a system able to:

- train lead-specific networks (DAEs) able to capture and extract features from data that are specialized on the different leads so to optimize the feature extraction phase; taking advantage from this, this phase lead us also to compress ECG input data;
- based on the useful features extracted previously, train RNNs able to classify the different beats in a lead-specific manner but also to take into account for the temporal correlation between data (using the recurrent GRU units)

The rationale behind this algorithm is that we wanted to implement a system that was able to combine the DAEs compression and denoising abilities on raw data, extract optimal lead-specific features and classify the different beats based on them in a recurrent manner: in this way, our system is capable to take into account, in the classification process, both the spatial and the temporal correlation between beats, reducing at the same time memory and energy consumption.

IV. Signals and Features

The ECG heartbeat signals are obtained from PhysioNet St Petersburg INCART 12-leads Arrhythmia open source dataset.

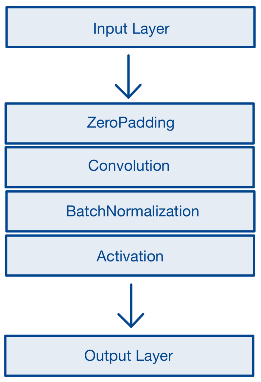


Fig. 1: High level description of the proposed CRNN

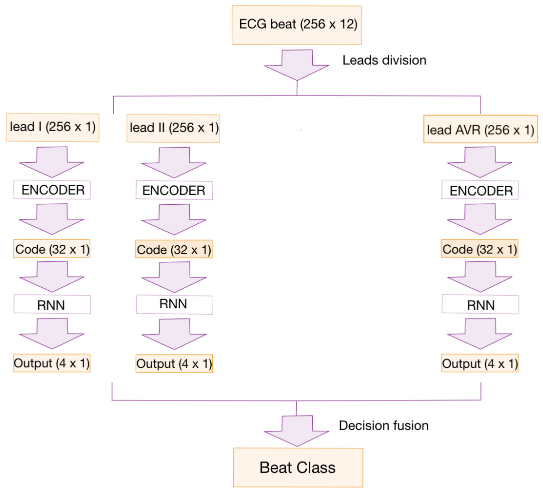
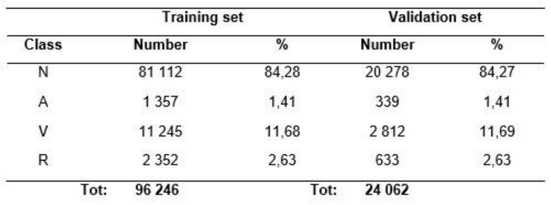


Fig. 2: High level description of the proposed ARNN

This database contains 75 annotated recordings extracted from Holter records, coming from 32 patients undergoing tests for coronary artery disease (17 mens and 15 womens, aged 18-80, mean age of 58). For each record, ”annotations” that indicate the times of occurrence and types of each individual heart beat (”beat-by-beat annotations”) are available; these ones were produced by an automatic algorithm, following the standard PhysioBank beat annotation definitions. The ECG records last 30 minutes and all the 12 leads are sampled at a frequency of 257 Hertz. From this records, over 175’000 heartbeats can be obtained, but for our investigation we considered only the first 50 records: we had so a database made by 120’338 beats. As before specified, each ECG record is related to a particular number of beats associated to a label. Following the standard PhysioBank beat annotation, we take into account 4 classes of beats: N (Normal beats), V (Premature Ventricular contraction), A (Atrial premature beat) and R (Right bundle branch block beat). In order to extract beats from the ECG recordings, input data went through a preprocessing segmentation phase, following this procedure: we used the positions indicated by the beat annotations available in the dataset and considered 120 samples before and 136 samples after the current annotation; this method was used also in other works like in [17]. Segmentation was done in all the leads for every heartbeat so each of them can be described with a data structure of dimensions (256,12). Moreover, before feeding data into the two networks, each lead was individually normalized between 0 and 1, to avoid that their different amplitude could contaminate feature extraction and classification process. No noise removal phase was implemented, as we decided to use ”raw” ECG signal to better simulate real acquisitions process conditions, that are inevitably affected by interference and noise. The overall obtained input data, composed by 120’338 beats were randomly splitted in training and validation set, for the homonym training and validation phase of classification process; in fact, using same data both for training and evaluation of an ANN likely lead to overfitting, meaning that the model perform very well with training data but not with newer ones; in the present work, training set was composed by 96246 (80% of the original dataset) beats and validation set by 24062 (20%) beats; despite the random selection, the proportion between classes were respected in both sets, as stated in the figure [3].



	Training set		Validation set	
Class	Number	%	Number	%
N	81 112	84,28	20 278	84,27
A	1 357	1,41	339	1,41
V	11 245	11,68	2 812	11,69
R	2 352	2,63	633	2,63
Tot:	**96 246**	Tot:	**24 062**	

Fig. 3: Training and validation set composition

V. Learning Framework

In this section we are going to explain how the learning process is carried out by the two different networks exposing the mathematical rationale that characterizes the different networks blocks but also the learning and validation process.

5.1 First Network: CRNN model

This network is fed with a matrix of size (12x256) for each beat and it is composed by 7 convolutional layers, 3 pooling layers, 1 GRU and 1 final dense state. Its structure is illustrated in fig.[4].

5.1.1 CNN

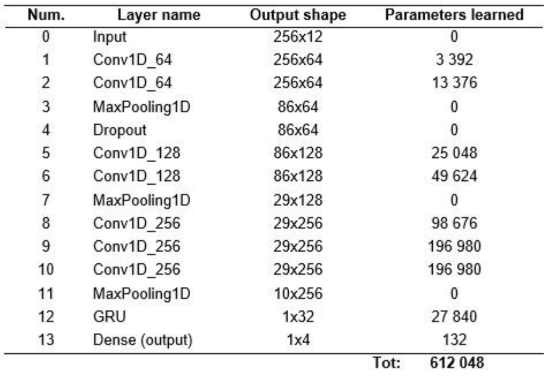
The convolution, that occurs between input and kernel, in each specific layer, is described by the following formula:

$$x_n = \sum_{k=0}^{N-1} y_k f_{n-k}$$

Where y, N e f are the input, the number of element of y (256 in our case) and the kernel, respectively. The latter, of size (3x12), is translated along the temporal dimension of the input with stride equal to one; in this way it is centered in each time step. The number of kernels, called layers depth, is the only parameter that varies between the convolutional layers: for the first two is 64 and it is doubled after each pooling layer. These parameters were chosen following the basic principles of VGGNet proposed by [9]:

1) applying more small kernels, instead of fewer and large ones, is useful for reducing the number of parameters;
2) the number of nodes, and thus the application number of activation function (rectified linear unit (ReLU)), increase;

This fundamentals makes the approach more discriminative. A ZeroPadding precedes all convolutional layers. This operation adds a line of zeros at the top and bottom of the matrix to avoid the change of size caused by the convolution. Only the MaxPooling layers, with kernel of size 3 and stride 3, reduce the input dimensions by a factor of $\left(\frac{1}{8}\right)$. Layer details and network parameters can be found in figure 4.



Num.	Layer name	Output shape	Parameters learned
0	Input	256x12	0
1	Conv1D_64	256x64	3 392
2	Conv1D_64	256x64	13 376
3	MaxPooling1D	86x64	0
4	Dropout	86x64	0
5	Conv1D_128	86x128	25 048
6	Conv1D_128	86x128	49 624
7	MaxPooling1D	29x128	0
8	Conv1D_256	29x256	98 676
9	Conv1D_256	29x256	196 980
10	Conv1D_256	29x256	196 980
11	MaxPooling1D	10x256	0
12	GRU	1x32	27 840
13	Dense (output)	1x4	132
		Tot:	**612 048**

Fig. 4: CRNN layers details and network parameters

5.1.2 RNN

For the implementation of this model we used a single GRU layer. As mentioned before, this layer, that receive as input the output of CNN, allows a flow of information along time axis. In other words, the current output is determined using the information from current input and the state of GRU at previous iteration: this can be achieved using a recursive formulation that lead to a Directed Acyclic Graph (DAG) structure. The following equations and figure [5] can help to understand how GRU units work:

$$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}^{(z)}\boldsymbol{x}^{(t)} + \boldsymbol{U}^{(z)}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}^{z})$$
$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}^{(r)}\boldsymbol{x}^{(t)} + \boldsymbol{U}^{(r)}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}^{r})$$
$$\tilde{h}^{(t)} = \tanh(\boldsymbol{W}^{h}\boldsymbol{x}^{(t)} + \boldsymbol{U}^{h}(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}) + \boldsymbol{b}^{h})$$
$$\boldsymbol{h}^{(t)} = \boldsymbol{z}^{(t)} \odot \boldsymbol{h}^{(t-1)} + (1 - \boldsymbol{z}^{(t)})\tilde{h}^{(t)}$$

In particular, $\boldsymbol{h}$ and $\boldsymbol{x}$ represent respectively the input and the output layer; these are combined, with different weights, to obtain $\boldsymbol{z}$ and $\boldsymbol{r}$: the output of the update gate and the reset grate. The latter is used to limit the influence of the previous state in the estimate of $\tilde{h}$. Moreover, $\boldsymbol{z}$ is applied in the final state to compute the new one. Intuitively, we can see its regulating actions: with high value of $\boldsymbol{z}$, the current state $\boldsymbol{h}^{t}$ is going to be influenced more by the previous one $\boldsymbol{h}^{(t-1)}$, whereas, for lower values, $\tilde{h}^{t}$, and so the input, weight more; $\sigma(\cdot)$ and $\tanh(\cdot)$ represent the two activation functions sigmoid and hyperbolic tangent respectively; $\odot$ is indicating an element-wise product between vectors. In out architecture, the GRU layer has a size of (32), so it also perform a dimensional reduction of the input. The last layer of the network is a common dense layer composed of four nodes, one for each class we are classifying. The states of the nodes are determined by combining all the information from the previous layer using appropriate weights. The results is the input of activation function, that for classification purposes is Soft-Max:

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \frac{\exp(z_k(\boldsymbol{x}, \boldsymbol{w}))}{\sum_j \exp(z_j(\boldsymbol{x}, \boldsymbol{w}))}$$

y_k represent the output of the k-th node and indicates the probability that the input beat belongs to the class k ($\sum_k y_k(\boldsymbol{x}, \boldsymbol{w}) = 1$). The beat will be assigned to the class with great probability.

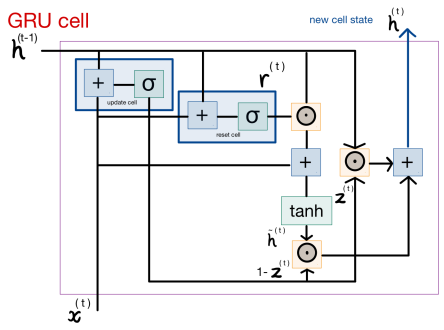


Fig. 5: Gru unit structure

5.2 Second Network: ARNN model

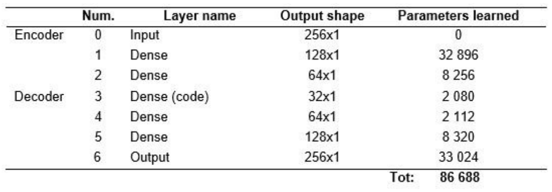
This second type of architecture, as specified in Section III, can be suddivided mainly in three part:

1) Feature extraction, carried out by 12 Deep Autoencoders working in parallel specifically on the single leads of the considered beat;
2) Classification, made by 12 Recurrent Neural Networks operating also in parallel on the single leads;
3) Decision fusion step, where the final label assigned to the beat is calculated taking the most probable label between the classification results on all the 12 or in a subgroup of leads;

Now we are going to explain in details the mathematical transformations applied by the different networks blocks on input data, keeping in mind that, for the first and second phase, the input is a single lead of a single beat (fig.[7]) and there are 12 DAEs and RNNs (12 ARNN units that forms the whole ARNN-12 network) processing data in parallel on every single beat lead (fig.[2]). Details about network layers can be found in figure [6].

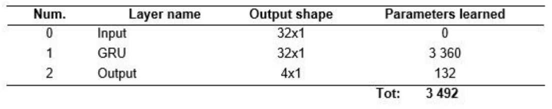
5.2.1 Deep Autoencoder (Feature Extraction phase)

Deep Autoencoders are a particular type of Feed-Forward Neural Network (FFNN): in our algorithm, these networks are used as "feature extraction" and dimensionality reduction tools in order to compress the original input data and retain only the useful properties of the input that are then send to the second classification step; from a more general point of view,



	Num.	Layer name	Output shape	Parameters learned
Encoder	0	Input	256x1	0
	1	Dense	128x1	32 896
	2	Dense	64x1	8 256
Decoder	3	Dense (code)	32x1	2 080
	4	Dense	64x1	2 112
	5	Dense	128x1	8 320
	6	Output	256x1	33 024
			Tot:	86 688

(a) *Deep Autoencoder network and model parameters*



Num.	Layer name	Output shape	Parameters learned
0	Input	32x1	0
1	GRU	32x1	3 360
2	Output	4x1	132
		Tot:	3 492

(b) *Recurrent Neural Network network and model parameters*

Fig. 6: ARNN layers and network parameters

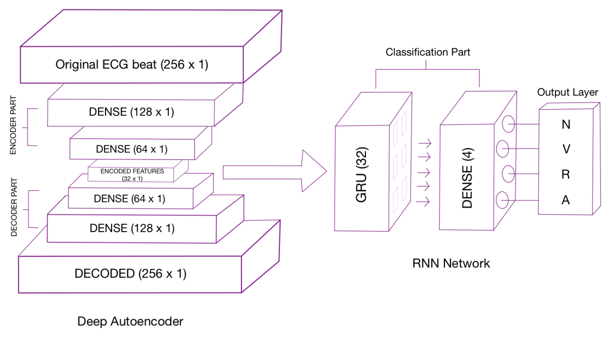


Fig. 7: Block representation of the ARNN network for one beat lead

Autoencoders, during the training step, learn a function that, thanks to the encoder part, is able to capture through several fully connected layers an hidden representation of the input data of lower dimension (”code”); this encoded representation is then used by the second ”decoder” part to reconstruct the original input data. From a general point of view, the network is composed by several layers of ”neurons” connected to each other so that each neuron is linked with all the neurons of the next layer. The first part of DAE is, as previously mentioned, the ”encoder” one: it begins with the so called ”input layer” that contains the input data fed into the network. In this case, the input was a list of 128’338 beats of size of (256,12), as 12 is the number of leads and 256 the number of data samples composing a single heartbeat: this list goes into a series of lead-specific Deep Autoencoders, so that each of them has input objects $\boldsymbol{x_j}$=$(x_{0j}$,...,$x_{nj})^T$ of size (256,1) ($\boldsymbol{x_j}$$\in$$\mathbb{R}^n$, where n=256)(fig.[7]). The input layer is then followed by four fully-connected (deep) layers (called ”hidden layers”): these are composed by a number of nodes (c_1,c_2,..,c_5) that apply a non linear activation function to a linear combination (made through a set of weights $\boldsymbol{w_{ji}}$) of the inputs coming from the previous layers; the set of weights are contained in the weight matrices $\boldsymbol{W}^1, \boldsymbol{W}^2, ..\boldsymbol{W}^5$ and are learned during the training phase; in the proposed model the activation function utilized is the so called ReLU (Rectifier linear unit) that is defined as: ReLU($\boldsymbol{x_j}$)=max(0,$\boldsymbol{x_j}$); thus, the generic output of a neuron belonging to one hidden layer, is defined as $o_i^{(l)}$=$f(W^{(l-1)}o_i^{(l-1)})$=$f(\sum_j w_{ji}^{(l-1)}o_j^{(l-1)})$, such that l is the layer number and $o_i \in \mathbb{R}^{c_i}$(where c_i is the number of neurons in that particular hidden layer). This operation is iterated from the second layer to the last one, the ”output layer”. The peculiarity of DAEs is that input and output layers have the same dimensions ((256,1)), whereas the deep hidden layers have lower dimensions c_i: in the encoder part, hidden layers dimensions are put in descending order ([$c_1, c2, c3$]=[128,64,32], as in figure [6]); as a result, the deepest one with the lowest dimension represents the ”encoded” version $\boldsymbol{x'_j}$ of the original input $\boldsymbol{x_j}$: here, $\boldsymbol{x'_j}$ have size (32,1), obtaining a data compression ratio of $\left(\frac{32}{256}\right)$=$\left(\frac{1}{8}\right)$. This codes represents our ”feature vector” and is the input of the second ”decoder” part; this block map back the encoded feature vector to a reconstruction $\hat{x}_j$ of the original input using a set of ascending number of weights ([$c_4, c5, c6$]=[64,128,256]); thus, the decoder is able to transform the encoded hidden representation into a newer one with the original data dimension that can be found in the last ”output layer”. Here the activation function is the Sigmoid $\sigma(o_i) = \left(\frac{1}{1+e^{-o_i}}\right)$; this function in our case revelead to be more robust during training process than other functions and lead our model to have the ability to capture multi-modal aspects of the input signal.

5.2.2 Recurrent Neural Network (Classification phase)

The classification phase is carried out by a series of 12 RNN working in parallel on the different leads, taking as inputs the features vectors of dimensions (32,1) created by the previously trained encoders and assigning them, through a learning process, the predicted class. As first layer of this second architecture (fig.[7]) there is a GRU unit (explained in section 5.1.2). Then, there is a fully-connected layer with four hidden neuron that, thanks to the activation function ”softmax”, is able to assign a label class to the ”single-lead” encoded beat. This second block is kept as simple as possible as the encoded input has small dimension and to maintain the overall computational cost low. Moreover, the classification output for the single lead of a beat is a vector of dimensions (4,1) that contains the ”probabilities” of the input to belong to one of the four classes: as a result, given the architecture structure, considering all the leads together every beat has associated a ”decision vector” with dimensions (4,12).

5.2.3 Decision Fusion phase

In order to assign a single label to the 12-lead beat, we decided to use a ”decision fusion” approach; given the ”decision vector” from the previous block of dimension (4,12), we summed up the probability values for the single label (summing along the first dimension) for each of the leads, obtaining a ”decision vector” of dimension (4,1) for the single beat; then, we normalized these values between 0 and 1 and assign the beat to the class with the highest probability value. Moreover, in order to assess if there are subgroups of leads that could be more useful than others for classification tasks, we decided to do the same computation but selecting a subgroup of 4 leads that, based on the confusion matrices produced by the previous classification step, had better classification results: in this case, the beginning "decision vector" had size (4,4).

5.3 Model training

5.3.1 CRNN training

The training process of CRNN was performed setting the batches at 128, the epochs equal to 128 and using Adam optimizer with a learning rate of 0.001. This is a very widespread optimizer characterized by a computational efficiency that well suits problems that are big in terms of data and parameters. Convolutional layers weights were initialized with He-normal initializer while GRU with orthogonal one. This shrewdness shows an increasing in convergence speed of parameter learning. The drop out rate, applied after the second layer, was fixed at 0.4, thus some nodes (40%) was fix at 0 to avoid overfitting during this phase. The so called "sparse categorical cross-entropy" is the loss function that was calculated at the end of every batch, looking at the computed output and the labels assigned to the beat. Then, the averaged gradient was backpropagated and the weight of all previous layers was updated.

5.3.2 ARNN training

Autoencoders are trained using an unsupervised learning algorithm where the target output values are set, despite traditional ANNs, equal to the input data: this procedure ends up in learning a set of weights that bring to a reconstructed output that, as much as possible, matches to the original input. The learning process follows a gradient descent method: network weights are updated in the direction of a negative gradient of a cost function J that in our case was "mean squared error":

$J = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_i||$

where N is the number of beats, $\hat{x}_i$ the decoder reconstruction and $\boldsymbol{x}_i$ the input. In this way we compute the weights considering the difference between the input and the reconstructed signal. Each of the twelve DAEs was trained individually on the specific lead for 50 epochs; batches size was set to 128 and Adam was chosen as network optimizer with a learning rate of 0.0001. The encoded versions of the leads of the beats belonging to the training set was utilized for the training of the final RNNs; the batch dimension was set at 32, whereas number of epochs and the optimizer remains the same of DAEs (learning rate in this case was fixed to 0.001). Since that the goal of the network was classification, we used "categorical cross entropy" as loss function.

VI. Results

Classification Performances

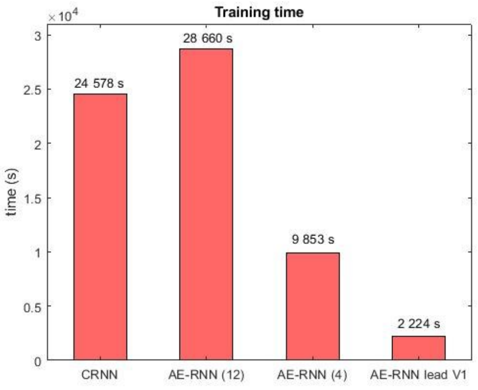
Confusion matrices and classification parameters obtained with CRNN and ARNN-12 (as before specified, ARNN module applied to all the 12 leads) on the validation set beats are shown in fig.[9(a)-(b)]. The first architecture CRNN has a slightly better classification performances, reaching an average accuracy of 99.90% ; moreover, sensibility and precision values are over 99% in 3 of the 4 considered beat classes; this fact indicates that the model is able to classify very well the beat to the right class. However, little lower values can be found for the class A; this in particular is a common trend that we found in all our trials; in our opinion this could be caused probably by the fact that:

- class A is the class with the fewest number of beats in the dataset; thus, the two ANNs are not able to update their weights in order to obtain good classification performances for this class;
- the particular beat morphology makes difficult the extraction of discriminant features;

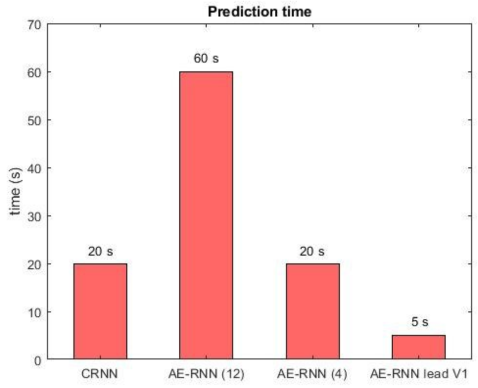
The second model ARNN-12 has different classification performances; in particular, the beat belonging to N class is slightly better classified with respect to the CRNN. Unfortunately, this seems to have undesired consequences: the number of false positive increase, causing a reduction of sensibility and precision for the other abnormal classes. The overall average accuracy on all the 12 leads reach almost 99.80%; however, we were not completely satisfied about this result. For this reason, we decided to focus on single lead classification performances(fig.[9]-(d)); from its inspection, we observed that the classification performances parameters have different trend on different leads and identifying a group of better (or worse) leads was possible. At least from the ANN point of view, this can be considered as a proof that the leads contain different information about beat physiology: some of them seems to be more informative and can generate more discriminant features, making easier their classification. Noticed this fact, we choose 4 leads with apparently better results (in particular, II,V2,V4,V5) and decided to merge, in the decision fusion phase, only the informations coming from them. The confusion matrix and the parameters obtained are reported in fig.[9]-(c). Comparing it with the previous result on all the 12 leads, we noticed almost negligible differences: in fact, in the second case, classification performances of class A increase, but a performance degradation occurs for class V; this imply almost equivalent overall average metrics. This particular result was unexpected to us because we thought that removing "bad" leads will enhanced performances: maybe they do not bring misleading information but simply not useful ones. More studies are so required about a possible optimal subset of leads.

Computational efficiency

One of the aims of our study was to satisfy the classification task taking also into account the computational cost of this operation, as it can be a discriminant factor on devices with limited power and resources. In particular, for the evaluation of computational efficiency, we considered training and validation time, as illustrated in fig.[8]. For the analysis of the second network, we made the assumption that the 12 ARNN units are working in series. From the acquired results, we can observe that:



(a) *Training time*

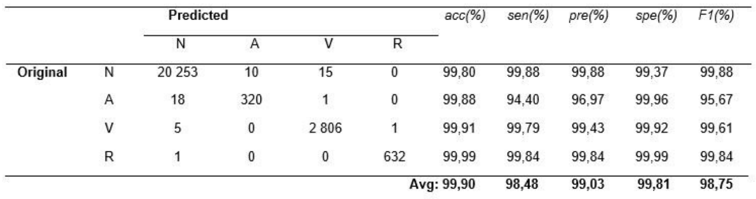


(b) *Prediction time*

Fig. 8: Training and prediction time of the two networks

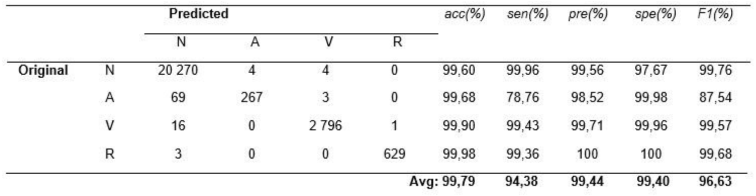
- CRNN has an high training time, because of the bigger number of parameters to learn, but in the prediction phase results to be very efficient;
- despite this, ARNN on 12 leads has a slightly higher overall training time with respect to CRNN (4000 seconds of difference, few more than 1 hour) but the prediction time was 3 times bigger;
- looking on single leads results (we considered lead V1 as an examples as it has a median total time), we observed a training time of 2224 seconds; this outcome was obtained from the sum of the single DAE (109 s) and the single RNN classifier (2115 s) training time, and a prediction time of 5 seconds, composed of 3 seconds of encoding and 2 seconds of classification;
- when we decrease the number of leads from 12 to 4, training time decrease and become much smaller than the one for CRNN, but prediction time remains the same;

Finally, because of the different classification performances on the different leads, it can be stated that each lead brings peculiar informations on cardiac activity; moreover, similar results with respect to the ARNN-12 ones can be obtained using also a subgroups of leads, abundantly reducing the overall computational cost: for this reason, more studies are necessary in order to investigate if there are strong links between the particular beat (or disease) and the features coming from specific leads.



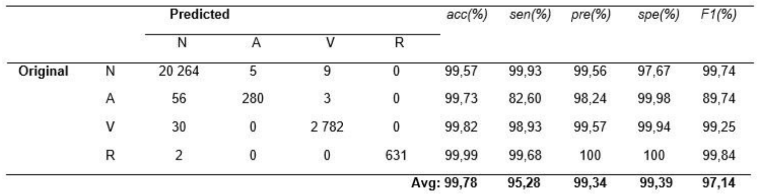
		Predicted				acc(%)	sen(%)	pre(%)	spe(%)	F1(%)
		N	A	V	R					
Original	N	20 253	10	15	0	99,80	99,88	99,88	99,37	99,88
	A	18	320	1	0	99,88	94,40	96,97	99,96	95,67
	V	5	0	2 806	1	99,91	99,79	99,43	99,92	99,61
	R	1	0	0	632	99,99	99,84	99,84	99,99	99,84
					Avg:	**99,90**	**98,48**	**99,03**	**99,81**	**98,75**

(a) *CRNN confusion matrix*



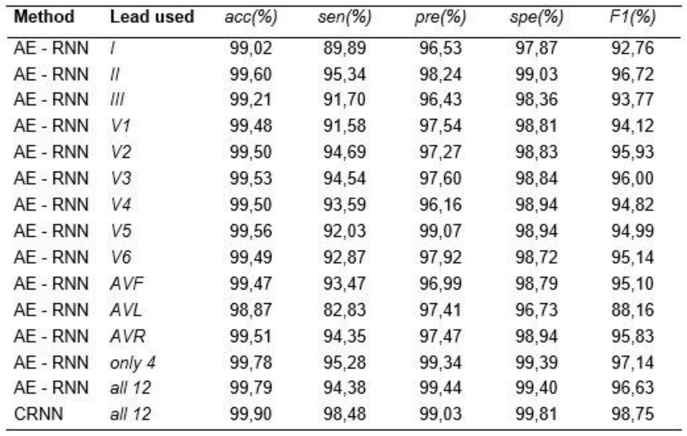
		Predicted				acc(%)	sen(%)	pre(%)	spe(%)	F1(%)
		N	A	V	R					
Original	N	20 270	4	4	0	99,60	99,96	99,56	97,67	99,76
	A	69	267	3	0	99,68	78,76	98,52	99,98	87,54
	V	16	0	2 796	1	99,90	99,43	99,71	99,96	99,57
	R	3	0	0	629	99,98	99,36	100	100	99,68
					Avg:	**99,79**	**94,38**	**99,44**	**99,40**	**96,63**

(b) *ARNN 12 leads confusion matrix*



		Predicted				acc(%)	sen(%)	pre(%)	spe(%)	F1(%)
		N	A	V	R					
Original	N	20 264	5	9	0	99,57	99,93	99,56	97,67	99,74
	A	56	280	3	0	99,73	82,60	98,24	99,98	89,74
	V	30	0	2 782	0	99,82	98,93	99,57	99,94	99,25
	R	2	0	0	631	99,99	99,68	100	100	99,84
					Avg:	**99,78**	**95,28**	**99,34**	**99,39**	**97,14**

(c) *ARNN best 4 leads confusion matrix*



Method	**Lead used**	acc(%)	sen(%)	pre(%)	spe(%)	F1(%)
AE - RNN	*I*	99,02	89,89	96,53	97,87	92,76
AE - RNN	*II*	99,60	95,34	98,24	99,03	96,72
AE - RNN	*III*	99,21	91,70	96,43	98,36	93,77
AE - RNN	*V1*	99,48	91,58	97,54	98,81	94,12
AE - RNN	*V2*	99,50	94,69	97,27	98,83	95,93
AE - RNN	*V3*	99,53	94,54	97,60	98,84	96,00
AE - RNN	*V4*	99,50	93,59	96,16	98,94	94,82
AE - RNN	*V5*	99,56	92,03	99,07	98,94	94,99
AE - RNN	*V6*	99,49	92,87	97,92	98,72	95,14
AE - RNN	*AVF*	99,47	93,47	96,99	98,79	95,10
AE - RNN	*AVL*	98,87	82,83	97,41	96,73	88,16
AE - RNN	*AVR*	99,51	94,35	97,47	98,94	95,83
AE - RNN	*only 4*	99,78	95,28	99,34	99,39	97,14
AE - RNN	*all 12*	99,79	94,38	99,44	99,40	96,63
CRNN	*all 12*	99,90	98,48	99,03	99,81	98,75

(d) *Single Leads performances*

Fig. 9: Classification performances of the evaluated networks

VII. Concluding Remarks

Heartbeat classification is still a challenging issue in the medical field and, due to the global relevance of CVDs, its resolution can be an important step in order to achieve a higher overall human health; for this purpose, ANNs can be an important and powerful tools that can lead to automatic cardiac disease diagnosis. In order to investigate deeper on the nature of ECG signals and neural networks, we decided to implements two networks: one with a CNN-RNN architecture, and the other one with a series of DAE-RNN working on specific leads. Looking at the overall performance, CRNN can achieve better classification results; in our opinion, this could be caused by the fact that one dimensional convolution approach is able to extract more discriminant features mixing the information from all the 12 leads simultaneously; on the other side, ARNN represent an approach that can be improved, not only to recognized more precisely the pathological beat, but also in the decision fusion phase; moreover, a ”feature fusion” approach, where the features coming from the DAEs layer are fused in a (32,12) encoded beat and followed by only one final RNN, could lead to better results; we tried to implement it, but without reaching some good results (maybe future studies on this approach could be useful). From a practical point of view, some observations can be reported; one of the possible pros of the separation of the feature extraction and classification phase (as in ARNN network), is the possibility to use a layer of pre-trained DAEs in order to compress data: this can be a discriminant factor in data management in devices with limited memory and capacity; also, the ARRN network is more flexible: one can easily ”switch on/off” the different leads and perform classification on the desired ones. Furthermore, our assumption on the ”serialization nature” of the second network could be not true on real application, where the possibility to compute operations in parallel can bring to different results. To conclude our project report, we wanted to present some final personal thoughts about what we learned; what we bring home is surely a lot of programming experience, since no one of the authors have never had any previous knowledge in Python coding; but despite implementations strugglings, we had the possibility to really touch with our hand how the all processes seen in theory at lectures really work, their huge potentiality, also characterized by their challenging application in real cases; we gained lot of what we can consider as ”soft skills” in managing the project ”road map” and code organization, learning new tools for document preparations like LaTeX: all things that can enhance our cultural baggage and surely augment our wealth of experience.

References

[1] H.Thomas, J.Diamond, A.Vieco, S.Chaudhuri, E.Shinnar, S.Cromer, P.Perel, G.Mensah, J.Narula, C.Johnson, G.A.Roth, A.E.Moran, Global Atlas of Cardiovascular Disease 2000-2016 The path to Prevention and Control, in Global Heart (New York,NY,USA,Geneva,Switzerland), Sep. 2018.

[2] Martis, Roshan Joy et al., Current methods in electrocardiogram characterization, in Computers in biology and medicine, vol. 48, 2014.

[3] PhysioBank Annotations, available at archive.physionet.org.

[4] R. Acharya, Shu Lih Oh, Yuki Hagiwara, J. H. Tan, M. Adam, A. Gertych, R. San Tan, A deep convolutional neural network model to classify heartbeats, in Computers in Biology and Medicine, vol. 89, 2017.

[5] T. Guo, J. Dong, H. Li and Y. Gao, Simple convolutional neural network on image classification, 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), (Beijing), 2017.

[6] L. Deng, G. Hinton and B. Kingsbury, New types of deep neural network learning for speech recognition and related applications: an overview, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, (Vancouver), 2013.

[7] Bo K Wong, Y. Selvi, Neural network applications in finance: A review and analysis of literature (1990-1996), in Information & Management, vol. 34, 1998.

[8] Qihang Yao, R. Wang, X. Fan, J. Liu, Y. Li, Multi-class Arrhythmia detection from 12-lead varied-length ECG using Attention-based Time-Incremental Convolutional Neural Network, in Information Fusion, vol. 53, 2020.

[9] Simonyan, Karen and Andrew Zisserman., Very deep convolutional networks for large-scale image recognition, 2014.

[10] Lu, Xugang, et al., Speech enhancement based on deep denoising autoencoder, in Interspeech 2013, Aug. 2013.

[11] L.Theis, W.Shi, et al., Lossy image compression with compressive autoencoders, 2017.

[12] D.Del Testa, M.Rossi, Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders, in IEEE Signals and Processing Letters, vol.22, Dec. 2015.

[13] O.Yildirim, R.S.Tan, U.R. Acharya, An efficient compression of ECG signals using deep convolutional autoencoders, in Cognitive Systems Research, 2018.

[14] A.Eduardo, H. Aidos, A.Fred, ECG-based Biometrics using a Deep Autoencoder for Feature Learning, An Empirical Study on Transferability, in Proceedings of the 6th International Conference on Pattern Recognition Application and Methods.

[15] O. Yildirim, P. Plawiak, R.S.Tan, U.Rajendra Acharya, Arrhythmia detection using deep convolutional neural network with long duration ECG signals, in Computers in Biology and Medicine, Sep. 2018.

[16] L. Gondara, Medical Image Denoising Using Convolutional Denoising Autoencoders, in 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), (Barcelona), 2016.

[17] S.Nurmaini, A.Darmawahyuni et al., Deep Learning-Based Stacked Denoising and Autoencoder for ECG Heartbeat classification, in Electronics 2020, Jan. 2020.

[18] O. Yildrim, U.B. Baloglu et al., A new approach for arrhythmia classification using deep coded features and LSTM networks, in Computer Methods and Programs in Biomedicine, 2019.

[19] Mikolov et al., Recurrent neural network based language model, in Eleventh annual conference of the international speech communication association, 2010.

[20] M.Zihlmann, D. Perekrestenko, M. Tschannen. , Convolutional recurrent neural networks for electrocardiogram classification, in 2017 Computing in Cardiology, 2017.