LSTM-Based ECG Classification for Continuous Monitoring on Personal Wearable Devices

Saeed Saadatnejad, Mohammadhosein Oveisi, and Matin Hashemi

This article is published. Please cite as: S. Saadatnejad, M. Oveisi, M. Hashemi, ”LSTM-Based ECG Classification for Continuous Monitoring on Personal Wearable Devices”, IEEE Journal of Biomedical and Health Informatics (JBHI), 2019.

***Abstract*—*Objective*: A novel ECG classification algorithm is proposed for continuous cardiac monitoring on wearable devices with limited processing capacity. *Methods*: The proposed solution employs a novel architecture consisting of wavelet transform and multiple LSTM recurrent neural networks (Fig. 1). *Results*: Experimental evaluations show superior ECG classification performance compared to previous works. Measurements on different hardware platforms show the proposed algorithm meets timing requirements for continuous and real-time execution on wearable devices. *Conclusion*: In contrast to many compute-intensive deep-learning based approaches, the proposed algorithm is lightweight, and therefore, brings continuous monitoring with accurate LSTM-based ECG classification to wearable devices. *Significance*: The proposed algorithm is both accurate and lightweight. The source code is available online [1].**

***Index Terms*—Continuous cardiac monitoring, Electrocardiogram (ECG) classification, Machine learning, Long short-term memory (LSTM), Embedded and wearable devices**

I. Introduction

CARDIOVASCULAR diseases (CVDs) such as myocardial infarction, cardiomyopathy and myocarditis are the leading causes of death in the world. An estimated 17.7 million people died from CVDs in 2015, representing 31% of all global deaths reported by the World Health Organization [2]. Cardiac arrhythmias are among the most important CVDs.

Electrocardiogram (ECG) signal represents electrical activities of the heart and is widely used in detection and classification of cardiac arrhythmias. A trained cardiologist can detect arrhythmias by visually inspecting the ECG waveform. However, arrhythmias occur intermittently, especially in early stages of the problem. Hence, it is difficult to detect them in a short time window of the ECG waveform. Therefore, continuous monitoring of patients' heartbeats in daily life is crucial to arrhythmia detection [3].

Wearable devices provide a platform for this purpose [3]. Our approach is to locally execute the ECG classification algorithm on patients' personal wearable devices. Local execution allows for continuous operation regardless of the network speed and availability. In addition, it allows data to stay on the wearable device and hence avoids privacy issues of cloud-assisted processing. Our approach is different from offline processing of stored ECG signals, or remote processing on powerful cloud servers [4], [5].

Authors are with the Learning and Intelligent Systems Laboratory, Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran. Webpage: http://lis.ee.sharif.edu, E-mail: saeedsa@ee.sharif.edu, oveisi@ee.sharif.edu, matin@sharif.edu (corresponding author).

Continuous monitoring on wearable devices require the automated ECG classification algorithm to be both *accurate* and *light-weight* at the same time. This forms our main focus in this work. Note that wearable devices have small and low-power processors which are much slower compared to desktop and server processors.

Many previous algorithms are based on morphological features and classical signal processing techniques [6]–[17]. Since the ECG waveform and its morphological characteristics, such as the shapes of QRS complex and P waves, significantly vary under different circumstances and for different patients, the fixed features employed in such algorithms are not sufficient for accurately distinguishing among different types of arrhythmia for all patients [18], [19].

To extract the features automatically and increase the heartbeat classification accuracy, deep-learning based algorithms including deep convolutional neural networks and recurrent neural networks have recently been proposed [19]–[23].

This paper proposes a novel ECG classification algorithm based on LSTM recurrent neural networks (RNNs). An overall view of the algorithm is shown in Fig. 1. The proposed algorithm employs RNNs because the ECG waveform is naturally fit to be processed by this type of neural network. The reason lies within the electrical conduction system of the heart which is shown in Fig. 2. The sinoatrial node generates a pacemaker signal which travels through internodal pathways to the atrioventricular node. The conduction slows through the atrioventricular node which causes a time delay. Next, the signal travels through the bundle of His to the heart apex and the Purkinje fibers, then finally to the ventricles [24]. The above sequence of electrical activities are reflected into the ECG waveform (Fig. 2), and therefore, temporal dependencies naturally exist in this waveform. RNNs capture such temporal dependencies in sequential data more efficiently compared to other types of neural networks.

As shown in Fig. 1, the proposed algorithm employs both LSTM recurrent neural networks and classical features, i.e., wavelet, at the same time. The additional features help to better capture the patterns in the ECG waveform. In addition, the proposed algorithm merges the arrhythmia predictions from small LSTM models as opposed to constructing one large model. See models α and β in Fig. 1. The total computational costs for executing multiple smaller LSTM models is lower than one larger LSTM model.

As a result, in contrast to many previous deep-learning based approaches which are computationally intensive [20]–

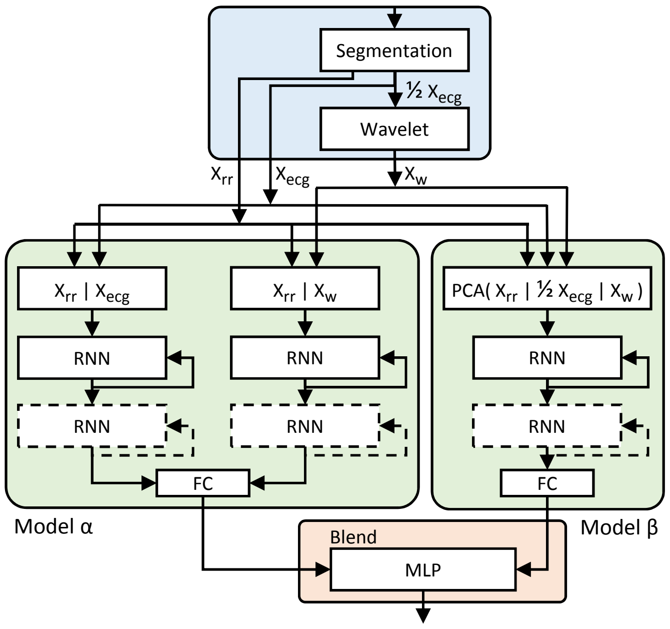


Fig. 1: Overall view of the proposed algorithm.

[23], the proposed algorithm increases the classification accuracy without significantly increasing the computational costs. Hence, it brings continuous monitoring with accurate LSTM-based ECG classification to personal wearable devices.

Experimental results show effectiveness of the proposed solution. Reporting the accuracy of ECG classification algorithms has been standardized by the Association for the Advancement of Medical Instrumentation (AAMI) [25]. Our proposed algorithm is evaluated using the same ECG signals that were employed in the previous works that conform to this standard. Experimental evaluations demonstrate that the proposed algorithm has superior classification performance compared to such methods. For instance, F_1 score is 3.3% and 15.5% higher in classifying ventricular ectopic beats (VEB) from non-VEBs and supraventricular ectopic beats (SVEB) from non-SVEBs, respectively. Note that SVEB detection is considered to be more difficult than VEB detection.

Computational requirements of the proposed algorithm is evaluated as well. Empirical measurements on small and low-power hardware platforms show that the proposed algorithm meets timing requirements for continuous and real-time execution on such platforms.

Previous works have lower classification performance [7]–[19], are not suitable for continuous execution on wearable devices due to high computational intensity [20]–[23], do not include all the standard AAMI classes [26]–[31], or focus on other problems related to processing of ECG signals [32]–[42].

Detailed comparisons with all the related works are presented in Section II. The proposed algorithm and its training procedure are discussed in Sections III and IV. The experimental results and discussions are presented in Sections V and VI. Concluding remarks are presented in Section VII.

II. Related Works

Many previous ECG classification algorithms are mainly focused on signal processing techniques including extraction of morphological features [7], frequency domain analysis [8], Hermite function decomposition [9], wavelet transform [10], [11], support vector machines [12] and hidden Markov models [13]. Hu et. al [14] proposed a mixture of experts method for patient-adaptable heartbeat classification. Chazal and Reilly [15] proposed a personalized heartbeat classification algorithm based on linear discriminant analysis on ECG morphology and timing interval features. Jiang and Kong [16] proposed a block-based neural network algorithm, and Ince et. al. [17] proposed particle swarm optimization for artificial neural networks, both for patient-specific heartbeat classification. Compared to the above solutions, the proposed algorithm achieves higher classification performance.

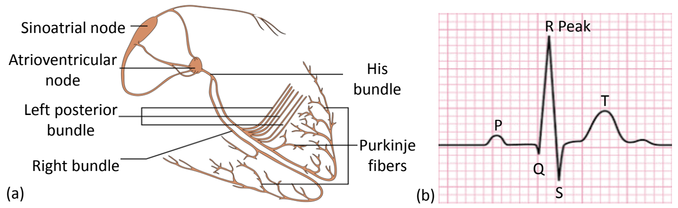


Fig. 2: (a) Electrical system of the heart. (b) ECG waveform [24], [43].

Recent approaches have focused on deep learning. Kiranyaz et. al. [19] proposed a one-dimensional convolutional neural network algorithm. Both the above and the proposed methods meet timing requirements, but our proposed method achieves higher classification performance, especially in SVEB detection. In [19], a heartbeat trio is fed into the network in order to capture the effect of nearby heartbeats in classifying the current heartbeat. This overhead is not necessary in our method since the LSTM cells capture temporal dependencies automatically and more efficiently. In addition, our proposed solution combines the arrhythmia predictions from small LSTM models as opposed to constructing one large model.

Rajpurkar et. al. [20] proposed a much deeper CNN. This computationally intensive algorithm is designed for a different problem namely classifying ECG signals into rhythms such as Sinus and Bigeminy. It consists of 34 layers and is not suitable for execution on wearable devices due to its very long execution time. It is about $10,000$X slower than the proposed method. Kachuee et. al. [21] proposed a deep CNN algorithm with 11 layers which is less accurate and about 100X slower compared to the proposed method. Jun et. al. [22] proposed another CNN algorithm with 11 layers but their convolutions are two-dimensional and hence much more computationally intensive than one-dimensional convolutions. In contrast, our proposed method is designed from ground up to be lightweight, and hence, meets timing requirements for continuous execution on wearable devices with limited processing capacity.

A general regression neural network was proposed in [23] for classification of long-term ECG signals. This algorithm is designed for offline processing and requires the entire recorded data. The algorithm is accelerated on high-performance GPUs in order to reduce the execution time. Teijeiro et. al. [6] proposed an ECG clustering algorithm that requires the entire recorded data. In contrast, our proposed algorithm is able to classify real-time ECG signals.

There are many other deep-learning based ECG classification methods in the literature that do not comply with AAMI standards and hence are not directly comparable with the proposed solution. For instance, many only consider a selected subset of the standard classes [26]–[31], which makes the design and training of neural networks much simpler because not all the challenging cases are included. The proposed solution fully complies with AAMI standards [25], the results are reported based on the standard and openly available MIT-BIH dataset [44] and all standard classification metrics have been calculated and reported.

Deep learning has also been applied to other problems related to analysis of ECG signals, for instance, ECG-based biometrics [32], [33], detecting atrial fibrillation (AF) [34]–[40] which is normally based on CinC Challenge 2017 dataset [41], and diagnosis based on hospital records [42].

III. Proposed Algorithm

Fig. 1 presents an overall view of the proposed algorithm. First, the incoming digitized ECG samples are segmented into heartbeats and their RR interval features and wavelet features are extracted (Sections III-A and III-B). Next, the ECG signal along with the extracted features are fed into two RNN-based models which classify every heartbeat (Sections III-C and III-D). The two outputs are then blended to form the final classification for every heartbeat (Section III-E).

A. Segmentation and RR Interval Features

The digitized ECG samples are segmented into a sequence of heartbeats. The segmentation is performed based on detecting the R peaks[1]. In specific, every segment (heartbeat) has a fixed length and contains 0.25 seconds of the input ECG signal before the detected R peak and 0.45 seconds after. This is denoted as X_{ecg} in Fig. 1.

R peak detection algorithms are well established and highly accurate. In our segmentation process, Pan-Tompkin's algorithm [45] is used. As part of Pan-Tompkin's algorithm, the time intervals between consecutive R peaks are calculated as well. Let RR_i denote the time interval from R peak $i-1$ to R peak i. Based on this information, we also extract the following four features for heartbeat i: I) RR_i as the past RR interval, II) RR_{i+1} as the next RR interval, III) $\frac{1}{10}\sum_{k=i-4}^{i+5} RR_k$ as the local average of the five past and the five next RR intervals, and IV) the average duration of the RR intervals in each person's train data. These four features are referred to as RR interval features, and form a feature vector denoted as X_{rr} in Fig. 1.

Note that features II and III require access to future heartbeats. However, as opposed to processing previously stored ECG signals, future information is not available in our setting. This is because the proposed algorithm is designed for continuous monitoring. We mitigate this problem by buffering the ECG signal in a first-in-first-out (FIFO) memory in real-time. This buffer is implemented in software and must be small. By always classifying the heartbeat which falls in the middle of this buffer, access to the near past and the near future information is made possible.

[1]R peak is a specific point in the ECG waveform as shown in Fig. 2(b).

The fourth feature is the average RR in each person's train data. Section IV-A discusses the train data. This feature varies among people with different average heart rates. For example, it is larger in athletes because they have slower heart rates.

Besides the above RR interval features which are accurately extracted with minimal computations, the proposed algorithm does not employ other hand-crafted morphological features such as those that are based on Q, S or T. This is because such features are not optimal in representing the characteristics of the underlying signal. In addition, they are fixed for all patients under all circumstances and therefore do not efficiently represent the differences among the arrhythmia classes [18], [19]. Instead, we let the features be automatically extracted using wavelet and recurrent neural networks as discussed in the following sections.

B. Wavelet Features

ECG signal has non-stationary characteristics. Therefore, to capture both the time and the frequency domain information, discrete wavelet transform [46] is applied to the digitized ECG samples in every heartbeat. In specific, Daubechies wavelet family is selected because of its similarity with the ECG signal [47]. Low-order Daubechies wavelets have high time resolution but low frequency resolution, while high-order ones have high frequency resolution and low time resolution [47]. Previous works mostly employed types 1 to 4. We employ type $T = 2$, i.e., db2, which falls somewhat in the middle of this range, and $L = 4$ levels of decomposition. Hence, the final list of wavelet coefficients is $X_w = (\mathrm{A4}, \mathrm{D4}, \mathrm{D3}, \mathrm{D2}, \mathrm{D1})$.

The computational complexity of discrete wavelet transform on an input array of size N with Daubechies type T and L levels of decomposition is

$$N \times T \times (1 + \frac{1}{2} + \ldots + \frac{1}{2^{L-1}}) \qquad (1)$$

Since the computational requirement is proportional to the input size, the wavelet input, i.e., X_{ecg} in Fig 1, is down sampled by a factor of 2 before applying the wavelet transform. This down sampling also cuts the total length of the wavelet output for every heartbeat, i.e., $|X_w|$, to about half, and thus helps to reduce the computational requirement of the following steps as well.

C. RNN-based Models

For every heartbeat, the input ECG samples (X_{ecg}) along with the extracted RR interval features and wavelet features (X_{rr} and X_w) are provided to two separate RNN-based models, called model α and model β. An overall view of the proposed algorithm is shown in Fig. 1. The two models make separate arrhythmia predictions which are then blended to form the final prediction for every heartbeat.

Employing X_{rr} and X_w in addition to X_{ecg} helps the RNN models capture patterns in the ECG signals more efficiently. The additional features provide processed information to the RNN models. Therefore, accurate results can be reached with

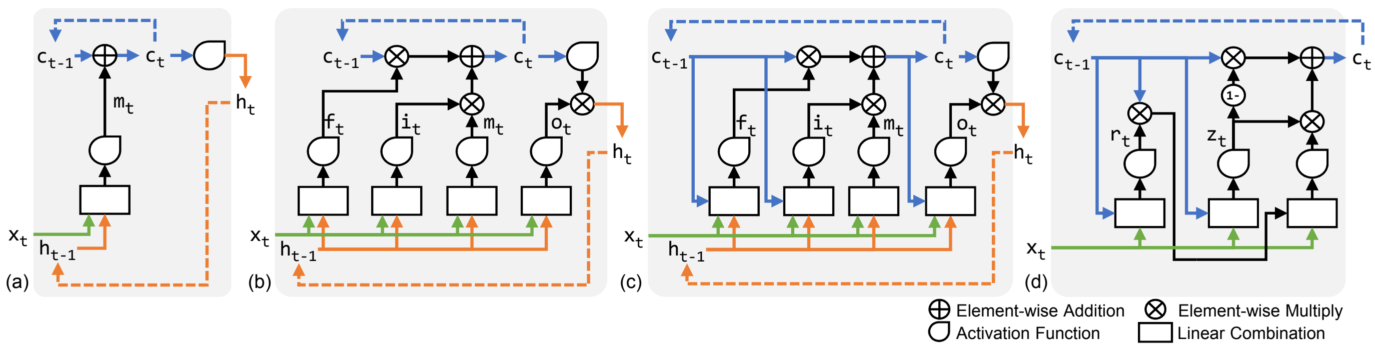


Fig. 3: (a) Simple RNN Cell, (b) Long Short-Term Memory (LSTM), (c) LSTM with Peepholes, (d) Gated Recurrent Unit (GRU).

smaller and hence faster RNNs. Employing multiple smaller RNNs in parallel instead of one larger RNN helps to increase the accuracy without significantly increasing the computational costs. The details are discussed in equation (11).

In model α, first X_w and X_{ecg} are processed separately and then the outputs are combined. In model β, however, X_w and X_{ecg} are first combined and then processed. The details are discussed below.

Model α: As shown in Fig. 1, model α consists of two branches. Every branch includes one or two RNNs. Every RNN includes a number of hidden units. The input to the left branch is denoted by $X^{\alpha 1}$ and is formed by concatenating X_{rr} and X_{ecg}. The RNN cells in this branch process the array $X^{\alpha 1}$ and extract $N_h^{\alpha 1}$ features. Similarly, the right branch concatenates X_{rr} and X_w into array $X^{\alpha 2}$, then, processes this array and extracts $N_h^{\alpha 2}$ features.

The outputs of the two branches are concatenated and fed into a fully connected neural network layer in order to produce the probability of all the N_y output arrhythmia classes. The output classes are discussed in Section V. The dimension of this fully connected layer is equal to $(N_h^{\alpha 1} + N_h^{\alpha 2}) \times N_y$. The maximum probability determines the arrhythmia class that is predicted by model α.

Model β: As shown in Fig. 1, this model consists of only one branch. As opposed to processing X_{ecg} and X_w in two separate RNN branches and then combining the outputs, here the inputs are combined. In specific, array X^β is formed by concatenating a down sampled version of X_{ecg} with X_{rr} and X_w, followed by applying PCA on the concatenated array. X^β is then processed by the RNNs in this model and N_h^β features are extracted. The features are fed into a fully connected neural network layer with dimension $N_h^\beta \times N_y$.

Models α and β include a number of hyper-parameters, namely, the number of RNNs in every branch, the number of hidden units in every RNN, and the RNN cell types. Hyper-parameter selection is discussed in Section IV-D. Different RNN cell types are discussed below.

D. RNN Cell Types

Simple RNN Cell: Fig. 3(a) shows a simple RNN cell. x_t is the input vector at time t. h_t and c_t are state vectors which are carried from time $t-1$ to time t, and hence, act as memory by encoding previous information. h_t is also considered as the cell output. Size of vectors h and c is denoted by N_h and is known as the number of hidden units. The cell works based on the following equations.

$$m_t[j] = \tanh\Big(\sum_{k\in[1,N_x]} w[j,k]x_t[k] + \sum_{k\in[1,N_h]} u[j,k]h_{t-1}[k] + b[j]\Big) \tag{2}$$

$$c_t[j] = c_{t-1}[j] + m_t[j] \tag{3}$$

$$h_t[j] = \tanh(c_t[j]) \tag{4}$$

As shown in (2), an intermediate vector m_t is formed by applying *tanh* activation function on a linear combination of x_t and h_{t-1}, i.e., current input and previous output, respectively. $j \in [1, N_h]$. Weight matrices w and u and bias vector b are determined during the training phase (Section IV). The state vector c_t is formed by accumulating m_t over time, as shown in (3). The output vector h_t is formed by applying *tanh* activation function on c_t. It can be seen that the output is related to all previous inputs.

Long Short-Term Memory (LSTM): In the above simple RNN cell the effect of all previous information is accumulated in the internal state vector. Gradient-based algorithms may fail when temporal dependencies get too long because gradient values may increase or decrease exponentially [48].

LSTM solves this issue by allowing to forget according to the actual dependencies which exist in the problem. The dependencies are automatically extracted based on the data. This is achieved through forget, input and output gates [48]. The LSTM cell is shown in Fig. 3(b). The gate signals are formed based on x_t and h_{t-1} as shown below.

$$f_t[j] = \sigma\Big(\sum_{k\in[1,N_x]} w_f[j,k]x_t[k] + \sum_{k\in[1,N_h]} u_f[j,k]h_{t-1}[k] + b_f[j]\Big) \tag{5}$$

$$i_t[j] = \sigma\Big(\sum_{k\in[1,N_x]} w_i[j,k]x_t[k] + \sum_{k\in[1,N_h]} u_i[j,k]h_{t-1}[k] + b_i[j]\Big) \tag{6}$$

$$o_t[j] = \sigma\Big(\sum_{k\in[1,N_x]} w_o[j,k]x_t[k] + \sum_{k\in[1,N_h]} u_o[j,k]h_{t-1}[k] + b_o[j]\Big) \tag{7}$$

In the above equations, σ denotes the *sigmoid* activation function, and $j \in [1, N_h]$. In the LSTM cell, m_t is computed as before, i.e., as in (2), but (3) and (4) are modified based on the forget, input and output gate signals as the following.

$$c_t[j] = f_t[j] \times c_{t-1}[j] + i_t[j] \times m_t[j] \tag{8}$$
$$h_t[j] = o_t[j] \times \tanh(c_t[j]) \tag{9}$$

As shown in (8), the forget gate f_t controls carrying of state vector c from time $t-1$ to time t. The input gate i_t adjusts the accumulation of m_t in c_t. As shown in (9), the output h_t is formed by applying $tanh$ activation function on c_t, and is then adjusted by the output gate o_t.

As the above equations show, the LSTM output still depends on all previous inputs. Previous information is neither completely discarded nor completely carried over to the current state. Instead, influence of the previous information on the current state is carefully controlled through the gate signals [48].

LSTM with Peepholes: The LSTM cell can be extended by adding extra connections from the internal state vector to the forget, input and output gates. The extra connections are marked with blue color in Fig. 3(c). The gate signals are formed based on a linear combination of x_t, h_{t-1} and now also c_{t-1} [49]. Detailed equations are omitted for brevity.

Gated Recurrent Unit (GRU): This cell is a simplified version of the LSTM cell which merges the two state vectors into one and also employs a different gating strategy [50]. The GRU cell is shown in Fig. 3(d). Here, c_t is formed as

$$c_t[j] = (1 - z_t[j]) \times c_{t-1}[j] \ + \ z_t[j] \times \tag{10}$$
$$\tanh\Big(\sum_{k\in[1,N_x]} w[j,k]x_t[k] + \sum_{k\in[1,N_h]} u[j,k]r_t[k]c_{t-1}[k] + b[j]\Big)$$

where, z_t and r_t are update and reset gate signals, respectively, and are formed similar to the LSTM gate signals as linear combinations of x_t and c_{t-1}.

Complexity Analysis: The above RNN cells perform several matrix and vector operations. For instance, the LSTM cell requires four matrix vector multiplications of size $N_h \times N_x$, four matrix vector multiplications of size $N_h \times N_h$ and several vector operations of size N_h. Total computational complexity for every execution of an RNN cell is therefore equal to

$$aN_xN_h + bN_h^2 + cN_h + d \tag{11}$$

where a, b, c and d depend on the cell type. According to the above equation, the computational complexity of an RNN cell has a quadratic growth with respect to the number of hidden units, i.e., N_h. Therefore, multiple smaller RNNs have lower computational costs in total compared to one larger RNN. For instance, the total runtime of two RNNs with $N_h = X$ is smaller than one RNN with $N_h = 2X$.

E. Blend Model

Ensemble methods such as blending are designed to boost the classification accuracy by blending the predictions made by multiple learning models [51]. As shown in Fig. 1, only two models are blended in our proposed algorithm in order to keep the computational requirement as low as possible. For every heartbeat, first, the two RNN-based models α and β independently compute the probability of all the N_y output arrhythmia classes. Then the two results are blended to form the final probability of the N_y output classes.

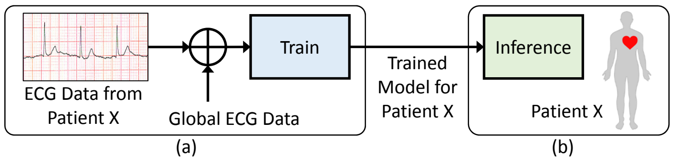


Fig. 4: (a) Patient-specific training. (b) Continuous ECG monitoring and heartbeat classification in real-time.

The blend model is implemented using a multi-level perceptron (MLP) with two hidden layers. The input and output layers have $2 \times N_y$ and N_y neurons, respectively.

IV. Training Procedure

A. Patient-Specific Training

We employ a patient-specific training procedure. In other words, the model is trained for every patient individually [14]–[17], [19]. Once the model is trained for a patient, continuous ECG monitoring and heartbeat classification is performed in real-time based on the trained model of that patient. This is shown in Fig. 4. Note that training is performed only once for every patient, i.e., it is not performed continuously.

As shown in Fig. 4(a), the training data for a patient is formed by combining two sets of data: local ECG data and global ECG data. The first part, i.e., local data, is specific to the patient and is helpful in increasing the classification accuracy due to existing similarities among the heartbeats of every patient. According to AAMI standards [25], this ECG data can be at most five minutes long. The second part, i.e., global data, is the same for all patients. It consists of a number of representative heartbeats from all arrhythmia classes. It helps the model learn other arrhythmia patterns that are not included in the local data. Details of the ECG signals employed in our experiments are presented in Section V.

Patient-specific training has been employed in [14]–[17], [19] as well. Another approach is to train only one model by feeding data from many patients, and then, use the trained model for classification of data from other patients. We do not employ this approach because the ECG waveform varies significantly among different patients [19].

B. Train the RNN Models

Back propagation (BP) is a well known method for training feed-forward neural networks such as convolutional neural networks (CNNs). This method cannot be applied to RNNs because of the existing temporal dependencies in the model, i.e., the feedback loops in Fig. 3 which carry previous information through time.

To train RNN models, train data is split into batches of several heartbeats each. The heartbeats are processed sequentially as the following. The weights are updated upon completion of every batch. In the beginning of every batch, h is set to zero and c is set randomly. Then the input data is forward-propagated over the network, and error is calculated until the batch finishes. Next, the error is back-propagated over the unfolded network in time, the weight matrices change in all instances and their mean is set as the updated weight. This is repeated until all batches are processed. This method is known as back propagation through time (BPTT) [52]. The optimization method employed in our work is adaptive moment estimation algorithm (Adam).

C. Train the Blend Model

First, the two RNN-based models, i.e., model α and model β, are trained independently as discussed above. Next, their output arrhythmia predictions for the heartbeats in the train data are used to train the blend model, i.e., the multi-level perceptron in Fig. 1. The training is performed using back propagation (BP).

D. Hyper-Parameter Selection

Learning algorithms related to neural networks often involve hyper-parameters. There are a number of guidelines and recommendations for selecting the hyper-parameters [53]. For the RNN-based models α and β, we perform a grid search on the range of $1-2$ for the number of recurrent layers, i.e., the number of RNNs in every branch, and $10-200$ for the number of hidden units, i.e., $N_h^{\alpha 1}$, $N_h^{\alpha 2}$ and N_h^{β}. For the cell type, we consider simple RNN cell, LSTM, LSTM with peephole, and GRU. Note that hyper-parameter selection is performed independently for model α, model β and the blend model. We found that RNN models with the LSTM cell, $N_h^{\alpha 1}=30$, $N_h^{\alpha 2}=30$, $N_h^{\beta}=50$ and one recurrent layer achieve consistently strong results.

V. Experimental Results

A. Setup and ECG Data

The proposed algorithm is implemented in the Python language and TensorFlow [54] library. Our source code is available online [1].

MIT-BIH ECG arrhythmia database [44] is used to evaluate the proposed algorithm and compare its performance with previous works. Each record in this database has two leads. The first lead is modified limb lead II. The second one is modified lead V1 or in some cases V2, V4 or V5. Two or more cardiologists independently annotated each record. The database contains two sets of data, called DS100 and DS200. DS100 includes representative samples of the variety of ECG waveforms and artifacts that an arrhythmia detector might encounter in routine clinical practice. DS200 includes complex ventricular, junctional, and supraventricular arrhythmias and conduction abnormalities. Based on AAMI standards, the records that contain paced beats $(102, 104, 107, 217)$ are excluded [25].

Training procedure is discussed in Section IV-A. The model is individually trained for every patient. Two sets of data, namely local data and global data, are combined for training the model for every patient. Global train data is formed by randomly selecting representative heartbeats from all arrhythmia classes in DS100 records. Local train data is the first five minutes of a patient's record in DS200. This is in compliance with AAMI standards [25]. Test data is all the records in DS200. The first five minutes of all the records are skipped in the test data.

5 Labels	7 Labels	Heartbeat types
N	N	Normal beat, atrial escape beat, junctional escape beat
	L	Left bundle branch block beat
	R	Right bundle branch block beat
S	S	Atrial premature beat, aberrated atrial premature beat, junctional premature beat, supraventricular premature beat
V	V	Premature ventricular contraction, ventricular escape beat
F	F	Fusion of ventricular and normal beat
Q	Q	Paced beat, fusion of paced and normal beat, unclassifiable beat

TABLE I: Heartbeat classes.

(a)

Reference	N	L	R	S	V	F	Q
N	35950	0	5	23	18	44	0
L	5	3034	1	0	0	0	0
R	0	0	2783	1	0	0	0
S	672	81	1	1566	17	3	0
V	216	1	17	45	4470	59	0
F	33	0	0	1	46	532	0
Q	6	1	0	0	0	1	0

(b)

Reference	N	S	V	F	Q
N	41778	24	18	44	0
S	754	1566	17	3	0
V	234	45	4470	59	0
F	33	1	46	532	0
Q	7	0	0	1	0

TABLE II: Confusion matrix with (a) 7 and (b) 5 heartbeat classes.

B. Classification Performance

In our experimental evaluations, every heartbeat is classified into the seven arrhythmia classes that are shown in Table I. Based on AAMI standards [25], many previous works employ five class labels, namely, N, S, V, F and Q [16], [17], [19]. However, to have more resolution, we split class N into three classes by separating two conduction abnormalities known as left bundle branch block (L) and right bundle branch block (R). As shown in Table II(a), the proposed algorithm is able to distinguish L and R from N very efficiently. To compare the proposed algorithm with previous works, L and R are merged back into N as shown in Table II(b).

In order to report performance results for binary classification of ventricular ectopic beats (VEB) from non-VEBs and also supraventricular ectopic beats (SVEB) from non-SVEBs, four statistical metrics, namely, accuracy (Acc), sensitivity (Sen), specificity (Spc), and positive predictivity (Ppr) are extracted from the confusion matrix.

$$\mathit{Acc} = \frac{TP+TN}{TP+TN+FP+FN} \tag{12}$$

$$\mathit{Sen} = \frac{TP}{TP+FN} \tag{13}$$

$$\mathit{Spe} = \frac{TN}{TN+FP} \tag{14}$$

$$\mathit{Ppr} = \frac{TP}{TP+FP} \tag{15}$$

The terms TP, TN, FP and FN denote true positive, true negative, false positive and false negative in the binary classification, respectively. Since, increasing Ppr often decreases

		VEB						SVEB					
		Acc	Sen	Spe	Ppr	F1	G	Acc	Sen	Spe	Ppr	F1	G
Dataset A	Hu et al. [14]	94.8	78.9	96.8	75.8	77.3	77.3	N/A	N/A	N/A	N/A	N/A	N/A
	Proposed	99.3	96.0	99.8	98.3	97.1	97.1	98.6	75.2	99.9	99.8	85.8	86.6
Dataset B	Chazal et al. [15]	99.4	94.3	99.7	96.2	95.2	95.2	95.9	87.7	96.2	47.0	61.2	64.2
	Proposed	99.6	95.8	99.9	97.8	96.8	96.8	99.0	75.6	99.9	98.9	85.7	86.5
Dataset C	Jiang and Kong [16]	98.1	86.6	99.3	93.3	89.8	89.9	96.6	50.6	98.8	67.9	58.0	58.6
	Ince et al. [17]	97.6	83.4	98.1	87.4	85.4	85.4	96.1	62.1	98.5	56.7	59.3	59.3
	Kiranyaz et al. [19]	98.6	95.0	98.1	89.5	92.2	92.2	96.4	64.6	98.6	62.1	63.3	63.3
	Proposed	99.2	93.0	99.8	98.2	95.5	95.5	98.3	66.9	99.8	95.7	78.8	80.0

TABLE III: Comparing the proposed algorithm with previous works in binary classification of VEB and binary classification of SVEB. Dataset A for VEB classification is 200, 202, 210, 213, 214, 219, 221, 228, 231, 233 and 234. Dataset A for SVEB classification is the same records for VEB classification plus 212, 222 and 232 [14]. Dataset B is 100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233 and 234 [15]. Dataset C is 200, 201, 202, 203, 205, 207, 208, 209, 210, 212, 213, 214, 215, 219, 220, 221, 222, 223, 228, 230, 231, 232, 233 and 234 [16], [17], [19].

Sen and vice versa, F_1 and G scores are also calculated which combine Sen and Ppr as the following.

$$F_1 = \frac{2}{\frac{1}{Sen} + \frac{1}{Ppr}} \tag{16}$$

$$G = \sqrt{Sen \times Ppr} \tag{17}$$

Table III compares the proposed algorithm with previous works. In order to provide thorough and fair comparisons, we employ the exact same data as the previous works. The proposed algorithm cannot be directly compared with other methods that do not comply with AAMI standards or do not employ the openly available MIT-BIH database. To compare the proposed algorithm with the methods in [14] and [15] we consider datasets A and B, respectively. The main dataset that is also employed in [16], [17] and [19] is dataset C.

In both VEB and SVEB, the proposed algorithm achieves superior classification performance compared to previous works. In VEB detection, for instance, accuracy is always higher than 99%. It is 4.5%, 0.2% and 0.6% higher than the previous works in datasets A, B and C, respectively. F_1 score is much higher. It is 19.8%, 1.6% and 3.3% higher than the previous works in datasets A, B and C, respectively.

In SVEB detection, accuracy is 3.1% and 1.7% higher than the previous works in datasets B and C, respectively. F_1 score is 24.5% and 15.5% higher than the previous works in datasets B and C, respectively. Note that F_1 score is a more meaningful metric compared to accuracy.

C. Real-time Execution

Personal wearable devices have small and low-power processors which are much slower compared to desktop and server processors. Therefore, to meet timing requirements for continuous execution, the proposed heartbeat classification algorithm needs to have low computational intensity.

Note that it is only the inference (test) phase which is executed repeatedly in real-time and needs to meet timing requirements. The training phase is performed only once in the beginning. In this section, we experimentally evaluate the execution time of the test phase on the hardware platforms shown in Fig. 5(a). All these platforms have small and low-power processors. The Java language and Android Studio are used to implement the source code for Moto 360 which is an AndroidWear device, and the C language is used for the other two hardware platforms.

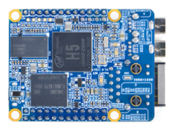
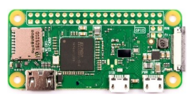
		Moto 360	NanoPi Neo Plus2	Raspberry Pi Zero
(a)	Device			
	Chipset	Snapdragon 400	Allwinner H5	Broadcom 2835
	CPU	ARM Cortex A7	ARM Cortex A53	ARM 1176
	Size	42 x 42 mm	40 x 52 mm	30 x 65 mm
(b)	Time	31.2 ms	39.1 ms	58.6 ms
(c)	Wavelet	5 %	5 %	3 %
	Model α	39 %	29 %	36 %
	Model β	53 %	63 %	57 %
	Blend	3 %	3 %	4 %

Fig. 5: a) Hardware platforms. b) Measured execution time. c) Distribution of the execution time.

Measured execution times are shown in Fig. 5(b). The proposed algorithm takes about 30 to 60 milliseconds to classify every heartbeat, while assuming a maximum heart rate of 200 bpm, a time window of at least 300 milliseconds is available. This shows that the proposed algorithm meets timing requirements for continuous ECG classification on small and low-power hardware platforms.

VI. Discussion

A. Ablation Study

In this section, different parts of the proposed model are modified and the results are experimentally studied. This helps to provide a better understanding of the impact of different parts of the proposed model.

No Wavelet: In order to experimentally study the effect of the wavelet features, we perform the same experiments described above but without the wavelet. Fig. 6 compares the results against the results of the original solution, i.e., the proposed solution. In specific, it shows the amount of degradation in the F_1 score.

We see that removing the wavelet features reduces the F_1 score by 5.1% and 8.3% for VEB and SVEB detection, respectively. This is because the wavelet transform provides processed information to the LSTM models, and thus, helps the models learn different patterns more efficiently. In addition, note that as shown in Fig. 5(c), the wavelet transform adds a very small overhead to the overall execution time.

Wavelet Types: We experiment with different wavelet types as well. Here db2 (the selected type) is replaced with db1, db3 and db4. As shown in Fig. 6, db2 and db3 which fall

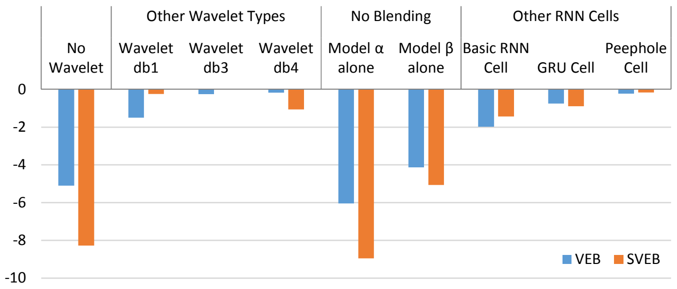


Fig. 6: Degradation of the F_1 score when different parts of the proposed algorithm are modified.

in the middle of this range yield the best performance, but db1 and db4 degrade the F_1 score. This is expected because higher time resolution is achieved by low-order types, while higher frequency resolution is achieved by high-order types. Between db2 and db3, we selected db2 because it employs a very small 4-point convolution kernel, and thus, provides a more computationally lightweight configuration.

RNN Cell Types: Next, the effect of using different RNN cell types is studied. In specific, the employed LSTM cell is replaced with simple RNN, GRU and Peephole cells. The results are shown in Fig. 6. The simple RNN and GRU cells degrade the classification performance, but the Peephole cell is very close to LSTM. We employ LSTM because it does not have the extra computations required in the Peephole cell, i.e., the extra blue connections in Fig. 3(c).

No Blending (Model α): In order to study the effect of combining two models, we perform the same experiments but with only one of the two models, in specific, model α. Hence, in addition to removing model β, the blend model itself is also removed. Fig. 6 compares the results of this experiment with the proposed solution in which models α and β are blended. When only model α is present, $F1$ score is 6% and 9% lower in VEB and SVEB detection, respectively. This shows that blending highly increases the classification performance.

No Blending (Model β): Similarly, when only model β is present, the classification performance is degraded. In specific, $F1$ score is 4.1% and 5.1% lower in VEB and SVEB detection, respectively.

Fig. 5(c) shows the distribution of the execution time in our hardware platforms. Model β has longer execution time compared to model α. This is because it has a larger LSTM cell. In specific, $N_h^{\beta} = 50$ but $N_h^{\alpha 1} = N_h^{\alpha 2} = 30$. As discussed in details in equation (11), two smaller LSTMs have lower computational costs in total compared to one larger LSTM.

B. Classification Performance with Limited Data

Single ECG Lead: The above experiments are based on two ECG leads. This is similar to previous works. However, in some wearable health monitoring devices, only one lead is available. Here the proposed algorithm is re-evaluated based on data from the first lead. The results are shown in Fig. 7. The proposed algorithm shows a relatively lower but still acceptable classification performance in this setting.

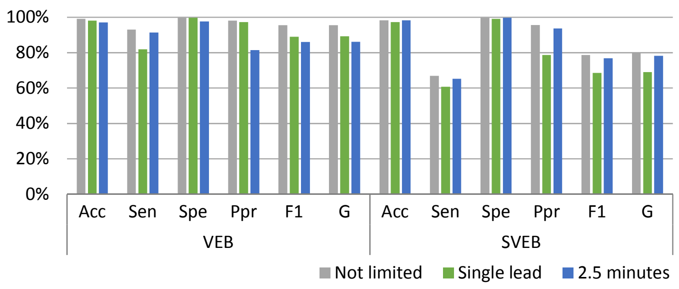


Fig. 7: Classification performance with limited ECG data.

2.5 Minutes ECG Data: Similar to previous patient-specific methods, the first 5 minutes of a patient's ECG record is used here as local data in the training phase. In practical settings, this data needs to be visually inspected and labeled by a specialist. Cutting the length of this data to half can help reduce the associated time and costs. Fig. 7 shows the classification performance of the proposed algorithm when local data is only 2.5 minutes long. The classification performance is relatively lower but still acceptable.

VII. Conclusion

In this paper a novel LSTM-based ECG classification algorithm was proposed which achieves superior classification performance compared to previous works. In addition, as opposed to many previous deep-learning based algorithms, it has low computational costs and meets timing requirements for continuous execution on wearable devices with limited processing power. Future directions include exploring other techniques to further increase the classification performance, studying other features in addition to wavelet, and improvements on single-lead ECG processing.

References

[1] Source code is available at http://lis.ee.sharif.edu/pub/2019_jbhi_soh
[2] "Cardiovascular diseases (CVDs)," May 2017. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs317/en/
[3] J. M. Bote, J. Recas, F. Rincn, D. Atienza, and R. Hermida, "A modular low-complexity ecg delineation algorithm for real-time embedded systems," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 2, pp. 429–441, March 2018.
[4] X. Wang *et al.*, "Enabling smart personalized healthcare: A hybrid mobile-cloud approach for ecg telemonitoring," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 3, pp. 739–745, May 2014.
[5] J. M. Lillo-Castellano *et al.*, "Symmetrical compression distance for arrhythmia discrimination in cloud-based big-data services," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 4, pp. 1253–1263, July 2015.
[6] T. Teijeiro, P. Flix, J. Presedo, and D. Castro, "Heartbeat classification using abstract features from the abductive interpretation of the ecg," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 2, pp. 409–420, March 2018.
[7] P. de Chazal, M. ODwyer, and R. B. Reilly, "Automatic classification of heartbeats using ecg morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1196–1206, July 2004.
[8] K. Minami, H. Nakajima, and T. Toyoshima, "Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 179–185, Feb. 1999.
[9] M. Lagerholm *et al.*, "Clustering ecg complexes using hermite functions and self-organizing maps," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 7, pp. 838–848, 2000.

[10] L.-Y. Shyu, Y.-H. Wu, and W. Hu, "Using wavelet transform and fuzzy neural network for vpc detection from the holter ecg," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1269–1273, July 2004.

[11] O. T. Inan, L. Giovangrandi, and G. T. A. Kovacs, "Robust neural-network-based classification of premature ventricular contractions using wavelet transform and timing interval features," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2507–2515, 2006.

[12] F. Melgani and Y. Bazi, "Classification of electrocardiogram signals with support vector machines and particle swarm optimization," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 5, pp. 667–677, September 2008.

[13] D. A. Coast *et al.*, "An approach to cardiac arrhythmia analysis using hidden markov models," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 9, pp. 826–836, September 1990.

[14] Y. H. Hu, S. Palreddy, and W. J. Tompkins, "A patient-adaptable ecg beat classifier using a mixture of experts approach," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 9, pp. 891–900, September 1997.

[15] P. de Chazal and R. B. Reilly, "A patient-adapting heartbeat classifier using ecg morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2535–2543, 2006.

[16] W. Jiang and S. G. Kong, "Block-based neural networks for personalized ecg signal classification," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1750–1761, November 2007.

[17] T. Ince, S. Kiranyaz, and M. Gabbouj, "A generic and robust system for automated patient-specific classification of electrocardiogram signals," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 5, pp. 1415–1426, May 2009.

[18] R. Hoekema, G. J. H. Uijen, and A. van Oosterom, "Geometrical aspects of the interindividual variability of multilead ecg recordings," *IEEE Transactions on Biomedical Engineering*, vol. 48, no. 5, pp. 551–559, May 2001.

[19] S. Kiranyaz, T. Ince, , and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, March 2016.

[20] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," *arXiv preprint arXiv: 1707.01836v1*, July 2017.

[21] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "Ecg heartbeat classification: A deep transferable representation," in *IEEE International Conference on Healthcare Informatics*, June 2018, pp. 443–444.

[22] T. J. Jun *et al.*, "Ecg arrhythmia classification using a 2-d convolutional neural network," *arXiv preprint arXiv:1804.06812*, 2018.

[23] P. Li *et al.*, "High-performance personalized heartbeat classification model for long-term ecg signal," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 1, pp. 78–86, Jan. 2017.

[24] How the Heart Works. [Online]. Available: https://www.nhlbi.nih.gov/health-topics/how-heart-works

[25] *AAMI-recommended practice: Testing and reporting performance results of ventricular arrhythmia detection algorithms*. Arlington, VA: Association for the Advancement of Medical Instrumentation, 1987.

[26] S. Chauhan and L. Vig, "Anomaly detection in ecg time signals via deep long short-term memory networks," in *Data Science and Advanced Analytics, IEEE International Conference on*, December 2015.

[27] A. Isin and S. Ozdalili, "Cardiac arrhythmia detection using deep learning," *Procedia Computer Science*, vol. 120, pp. 268–275, 2017.

[28] F. yan Zhou, L. peng Jin, and J. Dong, "Premature ventricular contraction detection combining deep neural networks and rules inference," *Artificial Intelligence in Medicine*, 2017.

[29] Ö. Yildirim, "A novel wavelet sequence based on deep bidirectional lstm network model for ecg signal classification," *Computers in biology and medicine*, vol. 96, pp. 189–202, 2018.

[30] S. L. Oh *et al.*, "Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats," *Computers in biology and medicine*, 2018.

[31] B. A. Teplitzky and M. McRoberts, "Fully-automated ventricular ectopic beat classification for use with mobile cardiac telemetry," in *IEEE International Conference on Wearable and Implantable Body Sensor Networks*, March 2018, pp. 58–61.

[32] A. Page, A. Kulkarni, and T. Mohsenin, "Utilizing deep neural nets for an embedded ecg-based biometric authentication system," in *IEEE Biomedical Circuits and Systems Conference*, 2015, pp. 1–4.

[33] R. Salloum and C.-C. J. Kuo, "Ecg-based biometrics using recuurent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2017.

[34] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," *Computing*, vol. 44, p. 1, 2017.

[35] P. Schwab *et al.*, "Beat by beat: Classifying cardiac arrhythmias with recurrent neural networks," in *Computing in Cardiology*, 2017, pp. 1–4.

[36] E. D. Ubeyli, "Combining recurrent neural networks with eigenvector methods for classification of ecg beats," *Digital Signal Processing*, vol. 19, pp. 320 – 329, 2009.

[37] ——, "Recurrent neural networks employing lyapunov exponents for analysis of ecg signals," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1192 – 1197, March 2010.

[38] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ecg features for screening paroxysmal atrial fibrillation patients," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–10, 2017.

[39] Y. Xia *et al.*, "Detecting atrial fibrillation by deep convolutional neural networks," *Computers in biology and medicine*, vol. 93, pp. 84–92, 2018.

[40] X. Fan *et al.*, "Multiscaled fusion of deep convolutional neural networks for screening atrial fibrillation from single lead short ecg recordings," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 6, pp. 1744–1753, Nov 2018.

[41] G. Clifford *et al.*, "Af classification from a short single lead ecg recording: the physionet computing in cardiology challenge 2017," *Computing in Cardiology*, vol. 44, 2017.

[42] Z. C. Lipton *et al.*, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

[43] Electrical conduction system of the heart. [Online]. Available: en.wikipedia.org

[44] R. G. Mark and G. B. Moody. (1997) MIT-BIH arrhythmia database. [Online]. Available: http://ecg.mit.edu/dbinfo.html

[45] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, pp. 230–236, May 1985.

[46] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674 – 693, July 1989.

[47] P. de Chazal, B. G. Celler, and R. B. Reilly, "Using wavelet coefficients for the classification of the electrocardiogram," in *Proceedings of the 22nd Annual EMBS International Conference*, July 2000.

[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[49] F. A. Gers and J. Schimidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, July 2000, pp. 189–194.

[50] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv: 1406.1078*, 2014.

[51] J. Sill *et al.*, "Feature-weighted linear stacking," *arXiv preprint arXiv:0911.0460*, 2009.

[52] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, no. 4, pp. 339 – 356, 1988.

[53] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[54] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283. [Online]. Available: www.tensorflow.org