Downloaded from orbit.dtu.dk on: Apr 25, 2024

DTU Library

A Deep Learning Approach for Real-Time Detection of Atrial Fibrillation

Andersen, Rasmus Sten; Peimankar, Abdolrahman; Puthusserypady, Sadasivan

Published in:
Expert Systems with Applications

Link to article, DOI:
10.1016/j.eswa.2018.08.011

Publication date:
2019

Document Version
Peer reviewed version

Link back to DTU Orbit

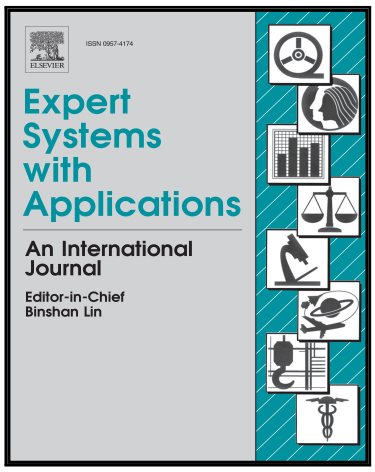
Citation (APA):
Andersen, R. S., Peimankar, A., & Puthusserypady, S. (2019). A Deep Learning Approach for Real-Time Detection of Atrial Fibrillation. *Expert Systems with Applications*, *115*, 465-473. https://doi.org/10.1016/j.eswa.2018.08.011

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Accepted Manuscript



A Deep Learning Approach for Real-Time Detection of Atrial Fibrillation

Rasmus S. Andersen, Abdolrahman Peimankar, Sadasivan Puthusserypady

PII: S0957-4174(18)30519-0
DOI: https://doi.org/10.1016/j.eswa.2018.08.011
Reference: ESWA 12141

To appear in: *Expert Systems With Applications*

Received date: 15 May 2018
Revised date: 20 July 2018
Accepted date: 9 August 2018

Please cite this article as: Rasmus S. Andersen, Abdolrahman Peimankar, Sadasivan Puthusserypady, A Deep Learning Approach for Real-Time Detection of Atrial Fibrillation, *Expert Systems With Applications* (2018), doi: https://doi.org/10.1016/j.eswa.2018.08.011

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Highlights

- Proposed a deep learning model for automatic detection of atrial fibrillation.
- An end-to-end model using CNN and RNN was developed to extract high level features.
- The model was trained and validated on three different publicly available databases.
- A post-processing scheme was also used to reduce the number of false positives.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

A Deep Learning Approach for Real-Time Detection of Atrial Fibrillation

Rasmus S. Andersen[a], Abdolrahman Peimankar[a], Sadasivan Puthusserypady[a,*]

[a]*Department of Electrical Engineering, Technical University of Denmark, Kongens Lyngby, 2800, Denmark*

Abstract

Goal: To develop a robust and real-time approach for automatic detection of Atrial Fibrillation (AF) in long-term electrocardiogram (ECG) recordings using deep learning (DL). *Method*: An end-to-end model combining the Convolutional- and Recurrent-Neural Networks (CNN and RNN) was proposed to extract high level features from segments of RR intervals (RRIs) in order to classify them as AF or normal sinus rhythm (NSR). *Results*: The model was trained and validated on three different databases including a total of 89 subjects. It achieved a sensitivity and specificity of 98.98% and 96.95% respectively, validated through a 5-fold cross-validation. Additionally, the proposed model was found to be computationally efficient and it was capable of analyzing 24 hours of ECG recordings in less than one second. The proposed algorithm was also tested on the unseen datasets to examine its robustness in detecting AF for new recordings which resulted in 98.96% and 86.04% for specificity and sensitivity, respectively. *Conclusion*: Compared to the state-of-the-art models evaluated on standard benchmark ECG datasets, the proposed model produced better performance in detecting AF. Additionally, since the model learns features directly from the data, it avoids the need for clever/cumbersome feature engineering.

Keywords: Electrocardiogram (ECG), Atrial Fibrillation, Deep Learning, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM).

1. Introduction

The prevalence of Atrial Fibrillation (AF) is increasing worldwide causing it to be one of the most important health issues in western countries (Zoni-Berisso et al., 2014; Fuster et al., 2006; Stewart et al., 2004). AF causes a significant decrease in quality of life and is a leading risk factor for stroke. Aditionally, AF is often associated with comorbidities such as hypertension, diabetes and heart failure (Stewart et al., 2004; Le Heuzey et al., 2004). AF requires long-term pharmacological treatment and hospitalisations resulting in a substantial and growing economic burden on the healthcare system (Fuster et al., 2006; Stewart et al., 2004; Le Heuzey et al., 2004).

The diagnosis of AF is based on history and clinical evaluation and requires electrocardiogram (ECG) documentation by at least a single-lead recording during the arrhythmia (Fuster et al., 2006). The fact that an ECG recording of the arrhythmia is a diagnostic criterion can make the process cumbersome, especially if the arrhythmia is paroxysmal and not easily provoked during a recording session. These cases suggest an ambulatory solution

*Corresponding author.
Email addresses: Rasmus.S.Andersen@dk.ey.com (Rasmus S. Andersen), apeima@elektro.dtu.dk (Abdolrahman Peimankar), spu@elektro.dtu.dk (Sadasivan Puthusserypady)

ACCEPTED MANUSCRIPT

(e.g., Holter monitoring) with extended recording time (+24 hours) to capture the arrhythmic event. Extended recordings from wearable ECG recorders introduce an infeasible amount of data for the physician to inspect and analyse and hence requires analytic software to automatically determine onset and duration of arrhythmic episodes. Signal processing and machine learning advocate for new and advanced methods for detection of arrhythmias, which could facilitate and accelerate the diagnostic process of AF.

Deep Learning (DL) have experienced a huge breakthrough in the last decade, beating state-of-the-art results in many applications such as machine vision and natural language processing (LeCun et al., 2015; He et al., 2016, 2015). The high capacity of DL models along with recent advances in parallel computing on Graphics Processing Units (GPUs) have allowed DL to outperform any other classification algorithm assuming the amount of data is sufficient. Because of their strong predictive capabilities, the application of DL model in biomedical signals is of keen interest. Up to now, most DL based ECG classification models have placed emphasis on classifying ECG beats. Kiranyaz et al. (2016) developed a 1-D convolutional neural network (CNN) real-time patient-specific ECG classification algorithm for the detection of ventricular ectopic beats and supraventricular ectopic beats with very high accuracy. Al Rahhal et al. (2016) proposed a DL approach for active classification of ECG signals which helps classifying the most difficult beats by adding them in the next training phase of the algorithm. Zubair et al. (2016) and Majumdar & Ward (2017) studied the possibility of beat-by-beat ECG classification into five major classes recommended by the Association for the Advancement of Medical Instrumentation (AAMI) using CNNs and greedy deep dictionary learning, respectively. Rajpurkar et al. (2017) developed a CNN algorithm for classifying ECG beats into fourteen different classes. In another study, Acharya et al. (2017) designed a deep CNN model for ECG beats detection into five common classes recommended by AAMI. Wu et al. (2016) used a deep belief networks to classify heartbeats into five classes. It should also be mentioned here that there is an interesting recent study by Mjahad et al. (2017) which reported a non-featured ECG arrhythmia classification.

In addition, there are other studies applying traditional machine learning pipelines using handcrafted feature extraction methods and traditional classification algorithms for ECG arrhythmia classification such as the ones reported in (Alonso-Atienza et al., 2012; Yu & Chou, 2008; Pawiak, 2018; Homaeinezhad et al., 2012; Shadmand & Mashoufi, 2016; Khalaf et al., 2015; Ceylan et al., 2009; Khorrami & Moavenian, 2010; Martis et al., 2012; Moavenian & Khorrami, 2010).

This study presents a novel DL model capable of detecting AF in long-term ECG recordings. The model learns features directly from the data and hence bypasses the need for feature engineering and prior domain knowledge. The proposed model outperforms state-of-the-art models evaluated on standard benchmark datasets and is computationally efficient allowing it to process 24 hours of recording in less than one second. In this paper, we investigate the performance of the proposed model on three different benchmark datasets. Additionally, the model is evaluated on heathy subjects with NSR to investigate any occurrence of false predictions.

The remainder of this paper consists of 6 sections. In Section 2, the theoretical background of CNNs and RNNs is briefly described. In Section 3, the components of the proposed model are introduced. The experimental results of the AF detection using the developed algorithm are presented and discussed in Section 4 and Section 5, and

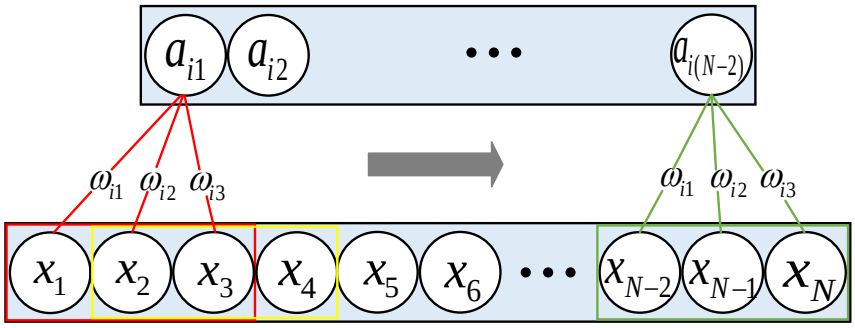


Figure 1: Temporal convolution. A receptive field of size $k_{size} = 3$ moving across a 1D time-series. The three corresponding weights (w_{i1}, w_{i2}, w_{i3}) for the ith filter are shown and the bias is left out for clarity. Here, N represents length of the input signals.

lastly, Section 6 concludes the paper.

2. Technical Background

Conventional machine learning methods have long been limited in their ability to process natural data in their raw form. Clever feature engineering is required to transform the data into a suitable internal representation from which the classifier can separate the classes (LeCun et al., 2015). DL breaks with this convention and beat state-of-the-art performance within many fields of machine learning without the need for prior domain knowledge and expertise. By stacking simple non-linear modules, DL methods are capable of extracting and classifying highly abstract features from raw data (LeCun et al., 2015; Nielsen, 2015).

In this study, two different types of DL network architecture are combined into a single classifier intended to detect and classify AF in long-term ECG recordings. The first part of the model consist of a multi-layer CNN, which extract features from the raw input sequence. The second part of the model uses a Recurrent Neural Network (RNN) structure known as Long Short-Term Memory (LSTM) to process the sequential features extracted by the CNN. Lastly, the output from the LSTM layer is passed to a single sigmoid neuron corresponding to a logistic classifier, which provides the posterior probability of input sequence containing AF. It is worth to note that the performance of the combined model can be enhanced through the training phase because both CNN and LSTM networks learn different functions. Therefore, a combination of these two networks achieves a higher classification accuracy (Geras et al., 2015; Shi et al., 2015; Sainath et al., 2015).

2.1. CNN Structure

Conventional neural networks, in which each neuron is connected to every neuron in the adjacent layer, is unable to take advantage of any spatial or temporal structure presented in the data (Nielsen, 2015). CNNs present a clever way of incorporating this information while simultaneously reducing the network complexity. Three key concepts lay the foundation of CNN: *receptive field*, *weight sharing*, and *pooling*.

ACCEPTED MANUSCRIPT

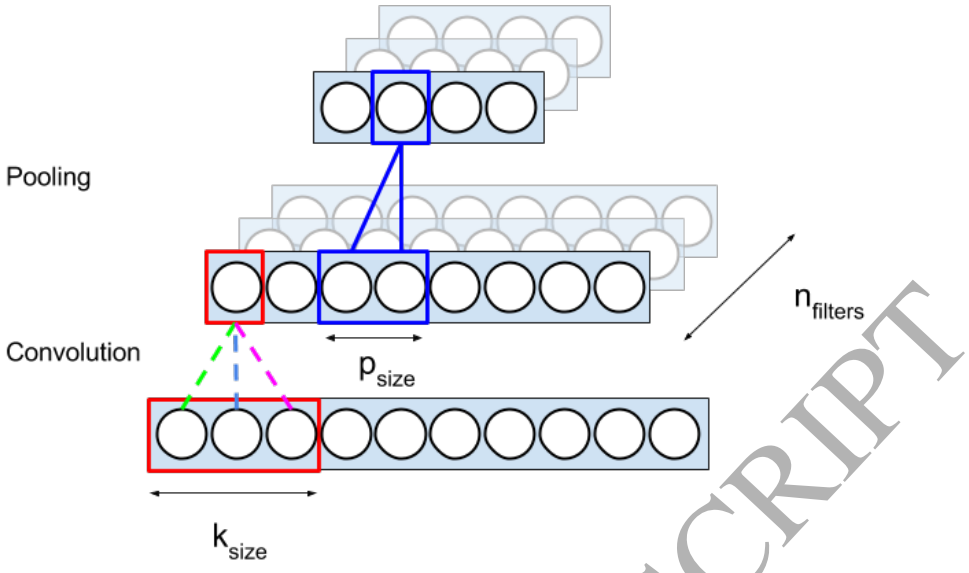


Figure 2: Convolutional network architecture with input layer, convolutional layer and pooling layer.

In CNNs, each neuron in the first hidden layer is connected only to a small region of the input neurons, known as the receptive field. Each connection learns a weight, and the neuron also learns an overall bias. Next, the window is slid across the entire input sequence, and each neuron in the hidden layer learns to analyse a specific part of the input sequence as shown in Fig. 1. The size or length of the receptive field is known as the kernel size k_{size}. Now instead of learning new weight and biases for each neuron in the hidden layer, the CNN learns only one set of weights and a single bias, which is applied to all neurons in the hidden layer. This is known as weight sharing. Mathematically, this can be expressed as:

$$a_{ij} = \varphi\left(b_i + \sum_{k=1}^{3} w_{ik} x_{j+k-1}\right) = \varphi\left(b_i + \mathbf{w}_i^T \mathbf{x}_j\right), \tag{1}$$

where a_{ij} is the activation or output of the j'th neuron of the i'th filter in the hidden layer, φ is the neural activation function, b_i is the shared overall bias of filter i, $\mathbf{w}_i = [w_{i1}\ w_{i2}\ w_{i3}]^T$ is vector with shared weight and $\mathbf{x}_j = [x_j\ x_{j+1}\ x_{j+2}]^T$. From a signal processing point of view, this can be expressed as convolution operation, where the input sequence, $\mathbf{x}_j$, is convolved with a filter with impulse response $\mathbf{w}_i$ (Goodfellow et al., 2016). Hence the output of the hidden layer is a filtered version of the input sequence and the learned weights corresponds to the impulse response of the filter. All neurons in the first hidden layer are hence trained to detect the same feature, just at different locations in the input sequence. Because of this property, the activations of the hidden layer is commonly referred to as a feature map (Nielsen, 2015).

To be able to detect more than a single localised feature, the network needs to compute additional feature maps. A complete convolutional layer hence consists of several feature maps. In Fig. 2, there is $n_{filters}$ different feature maps ($i = 1, \ldots, n_{filters}$), each defined by a set of three shared weights and a single shared bias. The resulting network can now detect $n_{filters}$ different kind of features, with each feature being detectable across the

ACCEPTED MANUSCRIPT

entire sequence (Nielsen, 2015).

Lastly, it is common to periodically insert a pooling layer in between successive convolutional layers (Nielsen, 2015; Goodfellow et al., 2016). The pooling layer simplifies the information in the output from the convolutional layer by sub-sampling. More specifically, the pooling layer summarises a region of e.g. $p_{size} = 2$ by returning the maximum value in that window. This is known as *max pooling* and is visualized in Fig 2. The moving window in a pooling layer typically uses strides of the same length as the pooling window compared to convolutional layers which normally uses a stride of 1. The output of a max-pooling layer with window size $p_{size} = 2$ is hence half the length of the input. Figure 2 provides a general overview of the CNN architecture with convolution and pooling. In modern literature, this structure provides the core component in deep convolutional models (Krizhevsky et al., 2012).

2.2. LSTM Structure

RNNs are another type of network architectures specifically designed to encompass sequential information. They are commonly used in sequence classification. The RNN structure, as opposed to a regular neural network, is capable of learning temporal dependencies and hence have shown superiority when working with time series data such as ECG signals. In traditional neural networks, all inputs are assumed to be independent of each other, but for most time-series and sequences this is not a valid assumption. RNNs work by performing the same task for every element in a sequence with the current output being dependent on previous computations hence the name *recurrent*. This study features an advanced type of RNN known as the LSTM introduced by Hochreiter and Schmidhuber in 1997 (Hochreiter & Schmidhuber, 1997). The LSTM network addresses the problem of unstable gradients and allows the network to learn long-term dependencies. In addition, it has been shown that the LSTM network outperforms other traditional RNN architectures (Graves & Schmidhuber, 2005).

The core idea behind the LSTM architecture is a continuously updated memory c_n. The LSTM memory block is illustrated in Fig. 3. It now has the ability to remove or add information to this memory at each time step in a sequence, carefully controlled by a forget gate f_n and an input gate i_n, which employ the same overall structure of a single layer neural network with a sigmoid activation function.

$$f_n = \varphi(b_f + \mathbf{u}_f^T \mathbf{x}_n + \mathbf{w}_f^T \mathbf{h}_{n-1}) \tag{2}$$

$$i_n = \varphi(b_i + \mathbf{u}_i^T \mathbf{x}_n + \mathbf{w}_i^T \mathbf{h}_{n-1}). \tag{3}$$

Here $\mathbf{x}_n$ is the input sequence at time step n, and $\mathbf{h}_{n-1}$ is the output vector from the LSTM at the previous time step. The parameters $\mathbf{u}_i$, $\mathbf{w}_i$, $\mathbf{u}_f$, and $\mathbf{w}_f$ are the input and recurrent weight vectors of the input and forget gates, respectively, and b's are all bias terms. The sigmoid activation function of the gates always return a value between 0 and 1 and hence control how much of each component should pass through.

The memory c_n is updated by partially forgetting the existing memory and adding a new memory content $\tilde{c_n}$

$$c_n = f_n c_{n-1} + i_n \tilde{c}_n, \tag{4}$$

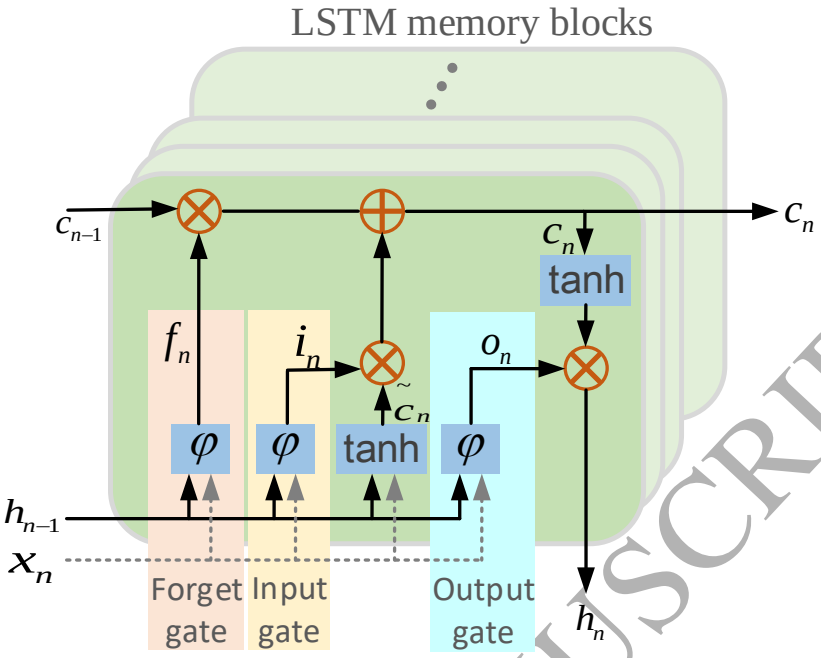


Figure 3: Architecture of a LSTM memory block.

where the new memory content is given by

$$\tilde{c}_n = \tanh(b_c + \mathbf{u}_c^T \mathbf{x}_n + \mathbf{w}_c^T \mathbf{h}_{n-1}). \tag{5}$$

The output is also gated by an output gate o_n of similar structure as the input and the forget gate. The output of the LSTM is hence given by

$$h_n = o_n \tanh(c_n), \tag{6}$$

where the output gate is computed as

$$o_n = \varphi(b_o + \mathbf{w}_o^T \mathbf{x}_n + \mathbf{u}_o^T \mathbf{h}_{n-1}). \tag{7}$$

The parameters $\mathbf{u}_o$ and $\mathbf{w}_o$ are the input and recurrent weight vectors of the output gate. Notice that the output gate is not only dependent on the input and previous output but also on the current memory. LSTM is able to decide whether to keep the existing memory or forget it via the introduced gates. Intuitively, this is an important property, as the network is capable of remembering features from early stages of a sequence and hence capture long-term dependencies (Chung et al., 2014).

One shortcoming of conventional LSTMs is that they are only able to make use of the previous context. If the network is not implemented in a real-time setting, there is no need not to exploit the future context as well (Graves & Jaitly, 2014). Bidirectional LSTMs (BiLSTM) do this by processing the data in both directions with two separate hidden layers, which are then fed forward to the same output layer. The output y_n is now a function

ACCEPTED MANUSCRIPT

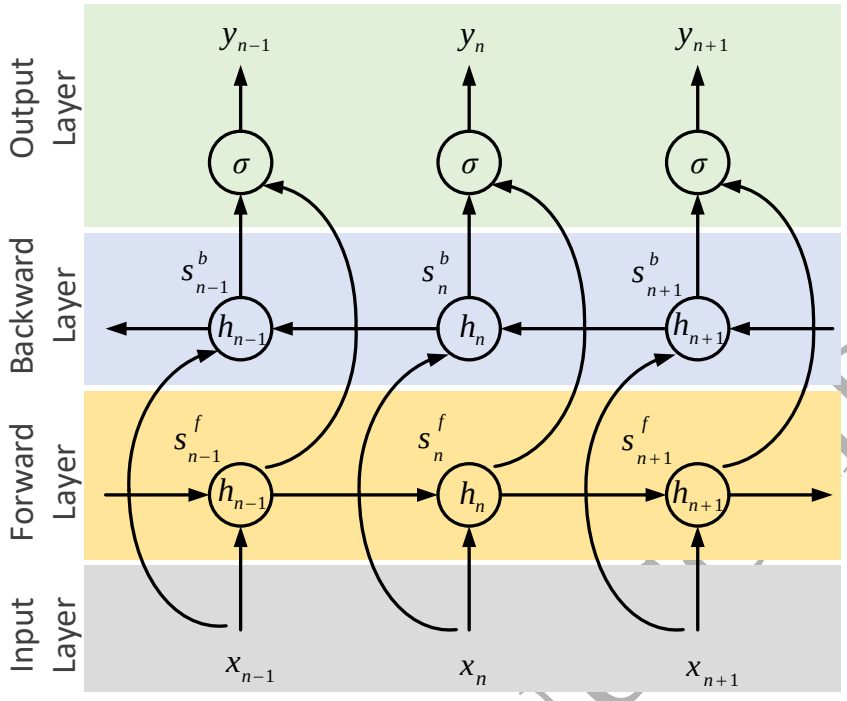


Figure 4: Illustration of an unfolded bidirectional LSTM. The input sequence is fed into two seperate hidden layers and processed in both directions before combining the two outputs in the output layer.

of the hidden state for the forward pass s_n^f and for the backward pass s_n^b along with the corresponding weights and biases as illustrated in Fig.4.

$$y_n = \sigma(w_f s_n^f + w_b s_n^b + b_h) \tag{8}$$

where w_f and w_b are the forward and backward weights, respectively, and σ represents the *softmax* function.

2.3. Training the Model

Choosing the parameters of the network (weights and biases) which achieve the optimal performance in a given task, is considered a non-convex optimisation problem, which requires a cost function ($\mathcal{J}$) to evaluate the performance of the network. Detection of AF is considered a binary classification problem and hence the binary cross-entropy (Eq. (9)) is used as cost function in this study.

$$\mathcal{J} = -\frac{1}{N_t}\sum_x (y \ln(a) + (1-y)\ln(1-a)), \tag{9}$$

where a is the activation of the output layer, y is the desired output and N_t is the total number of training inputs. Both a and y depends on the input x, but this is left out for notational simplicity.

The output of the network a is parametrised by all the weights and biases of the network and hence the cost function evaluates the performance of the network with regard to the trainable parameters. $\mathcal{J}$ is typically defined in a high dimensional space and contains non-linearities, which makes the optimisation non-convex (Nielsen, 2015).

ACCEPTED MANUSCRIPT

The optimisation problem is solved by employing an iterative scheme known as the Stochastic Gradient Descent (SGD).

3. Materials and Methods

3.1. Data

Three different databases (freely available from Physionet (Goldberger et al., 2000)) are used in the training and validation of the proposed model. They are the MIT-BIH AF Database (AFDB) (Moody & Mark, 1983), the MIT-BIH Arrhythmia Database (MITDB) (Moody & Mark, 2001) and the MIT-BIH NSR Database (NSRDB) (Goldberger et al., 2000).

The AFDB includes 25 long-term ECG recordings of human subjects with AF (mostly paroxysmal). The individual recordings are approximately 10 hours in duration and contain two-channel ECG signals each sampled at 250 samples/second with 12-bit resolution over a range of ± 10 millivolts (Goldberger et al., 2000; Moody & Mark, 1983). Each recording contains a beat annotation file prepared using an automatic R-peak detection algorithm. In this study, two of the 25 recordings (record *00735* and *03665*) have been excluded from further analysis as the signals are unavailable.

MITDB contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects (records 201 and 202 are from the same subject) studied by the BIH Arrhythmia Laboratory between 1975 and 1979. The recordings were digitised at 360 samples/second/channel with 11-bit resolution over a 10 mV range (Moody & Mark, 2001). Additionally, each recording contains an annotation file, with not only the location and type of each beat but also the onset and type of any arrhythmia in the recording (Moody & Mark, 2001; Goldberger et al., 2000). It is important to notice the split of recordings into the 100 series and the 200 series. The 100 series contains random samples of the population without any episodes of AF. Additionally, three of the recordings include paced beats (record 102, 104 and 107). The 200 series was manually selected to include less common arrhythmias such as ventricular bigeminy and trigeminy, along with 8 AF subjects (Moody & Mark, 2001; Goldberger et al., 2000).

Table 1: Overview of the three databases included in the study. The table shows the average record duration, the number of AF episodes, the average AF episode duration and the number of unique rhythms for each database. The number of R-peaks analyzed for each database is also reported in the last column.

Database	Record Duration (Hours)	AF Episodes	AF Duration (Seconds)	Unique Rhythms	R-peaks
AFDB	10.19	291	1155.4	4	828000
MITDB	0.50	33	16.2	15	86400
NSRDB	24.31	0	0.0	1	1555200

The NSRDB contains 18 long-term ECG recordings of subjects referred to the Arrhythmia Laboratory at Boston's Beth Israel Hospital (now the Beth Israel Deaconess Medical Center). Subjects included in this database were found to have no significant arrhythmias and is hence considered NSR (Goldberger et al., 2000). The recordings

were digitised at 128 samples/second/channel. Along with each digitised recording is a reference annotation file containing the location and type of each beat. Lastly, each database is summarised in Table 1.

3.2. Model Overview

DL offers a unique approach in which the features are learned directly from the input signal and hence no prior domain knowledge is required to engineer useful features. The novel algorithm proposed in this study is a multi-layer DL network featuring both convolutional and recurrent layers. An overview of the network architecture can be found in Fig. 5. Furthermore, a detailed structure of the different layers including their output dimensions is shown in Fig. 6.

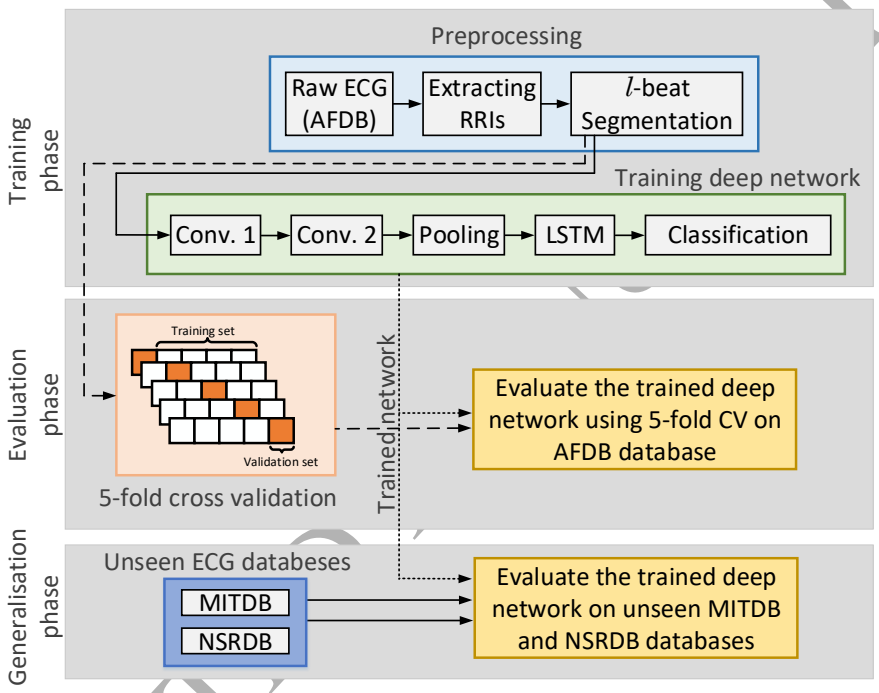


Figure 5: Flowchart of the proposed method. The method includes a training phase in which the optimal parameters for the network architecture is estimated, an evaluation phase for validating performance measures and a generalisation phase to report performance on previously unseen data sets.

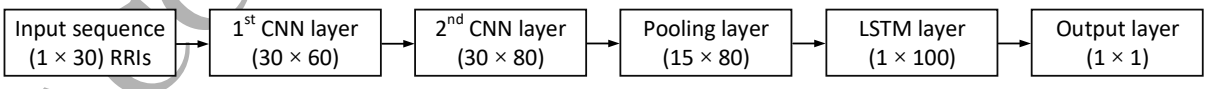


Figure 6: A detailed structure of the used layers in the DL model. The numbers in the parenthesis show the output dimension of each layer.

3.2.1. Preprocessing

The raw ECG recordings from the databases are converted to RRI sequences to reduce computational complexity and highlight AF behaviour. Measuring the interval between adjacent beats also known as RRI has shown good

ACCEPTED MANUSCRIPT

performance in detection of certain cardiac arrhythmias (Stein et al., 1994; Dash et al., 2009; Lake & Moorman, 2010; Colloca et al., 2013). The RRI sequence captures one of the hallmarks of AF, namely the irregular ventricular contraction rate. Furthermore, employing RRIs instead of raw ECG reduces the computational complexity, as only the information about R peak differences are stored as opposed to using all intermediate samples in the raw signal.

Converting the ECG recordings to RRI sequences requires segmentation of the ECG signals to obtain R peak locations. The R-peak detection algorithm is not a part of this study, and hence R-peak locations are obtained from the database annotation files. After locating the R-peaks, the RRI is calculated as:

$$RRI(n) = \frac{R_{peaks}(n+1) - R_{peaks}(n)}{f_s}, \tag{10}$$

where $R_{peaks}(n)$ is the location of the n'th R peak in terms of samples and f_s is the sampling frequency in Hz.

After extracting the RRIs for each recording in the databases, each RRI sequence is segmented into smaller signals of length l beats ($l = 31$), which is equal to 30 RRIs, used as input for the model. It should be also noted that the shift between two segments is equal to 10 beats. Because the model is now classifying l-beat segments, it is necessary to convert the beat-to-beat rhythm annotations into an l-beat resolution. This conversion is achieved by classifying any l-beat segment as true AF only if the number of true AF beats exceeds a predefined threshold T_{AF} (Dash et al., 2009), which is set to be 0.5 in this study. The resulting l-beat segments of RRI along with the corresponding binary label is used as the input and desired output, respectively.

3.2.2. Convolutional Layers

The input is fed directly into two successive convolutional layers, which extract temporal dependent features from the signal. The first convolutional layer has a kernel size of $k_{size,1} = 5$ and outputs $n_{filters,1} = 60$ features. Each feature is a filtered version of the input sequence, where the learned parameters of the layer correspond to the impulse response of the filter. To preserve the temporal dimension, the input is zero-padded before the convolution is applied. The input to the second convolutional layer is now a sequence of 60 features with same temporal dimension as the input. The second convolutional layer will extract more abstract features based on the features extracted in the first layer. The second convolutional layer uses a kernel size of $k_{size,2} = 3$ and outputs $n_{filters,2} = 80$ features. Once again the temporal dimension is preserved using zero-padding. The learned parameters of the second layer are not as easily interpreted as in the first layer because the input is now a 60-dimensional sequence. Instead, the parameters should be interpreted as tuned weights which produce the optimal features given the network architecture and data. The choice of kernel size for each layer is based on a coarse grid search of common values $k_{size} \in \{3, 5, 7, 10, 12\}$.

3.2.3. Pooling Layer

Following the two successive convolutional layers is a pooling layer. This layer performs the max-pooling operation using a kernel size of $p_{size} = 2$ with strides of two. The resulting output has half the temporal dimension of the input and hence reduces complexity for the following layer.

ACCEPTED MANUSCRIPT

3.2.4. LSTM Layers

The output from the pooling layer is fed into a bidirectional LSTM layer with $n_{units} = 100$ hidden units. The bidirectional setting of the LSTM layer enables it to learn long-range context in both input directions (Graves & Jaitly, 2014).

3.2.5. Classification

The output from the LSTM layer is fed into a single sigmoid neuron, as this is the non-redundant convention for binary classification in any neural network. The output of the sigmoid neuron is interpreted as the posterior probability of the i'th input sequence belonging to the AF class given the parameters of the model. The classification is now performed as follows:

$$\hat{y}_i = \begin{cases} 1 & \text{if } p(y_i = \text{AF}|\ \mathbf{x}_i,\ \text{MODEL}) \geq T \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $\hat{y}_i$ is the predicted class. The probability threshold T has a default value of 0.5 but can be altered to change the trade-off between the number of false positives and false negatives respectively.

3.2.6. Optimization

The network is trained using SGD with Nesterov accelerated gradient (NAG) (Nesterov, 1983). By applying NAG to the SGD scheme, it is possible to speed up the convergence and hence reduce the number of training epochs required to reach an optimum. On the contrary, the introduction of NAG will require fitting one additional hyper-parameter, the momentum coefficient μ. In practice, however, only a sparse set of values $\{0.999, 0.995, 0.99, 0.9, 0\}$ are considered, as these have shown to produce the best results in multiple situations (Sutskever et al., 2013).

The final network requires fine-tuning of 3 hyperparameters: the learning rate η, the momentum coefficient μ and the regularisation parameter λ for the $L2$ weight regularisation. Additionally, the use of dropout with different dropout probability p should be investigated throughout the network structure.

3.3. Evaluation Protocol

To evaluate the performance of the model, a number of standard statistical measures are employed. These feature: (i) Sensitivity (Se) defined as the ratio between True Positive (TP) segments and all positive segments; (ii) Specificity (Sp) defined as the ratio between True Negatives (TN) segments and all negative segments, (iii) Accuracy (Acc) defined as correctly classified segments divided by the total number of segments; (iv) False Positive Rate (FPR) defined as the ratio between False Positive (FP) segments and all negative segments; and (v) Positive Predictive Value (PPV) defined as the ratio of TP segments and all segments classified as positive. Lastly the Receiver Operating Characteristics (ROC) are used to visualise the trade-off between Se and FPR. The Area Under the Curve (AUC) is also used as a quantitative performance measure.

ACCEPTED MANUSCRIPT

3.3.1. Parameter Tuning

The 23 recordings from the AFDB is randomly split into a training set of 18 recordings and a test set of 5 recordings. The training set is again randomly divided into a training set of 14 recordings and a validation set of 4 recordings.The network is then trained on the training set and the performance is evaluated on the validation set. When the optimal setting is determined from the validation set, the model is trained on both the training and validation set before the final performance is evaluated on the test set. This setup assures that the reported performance is assessed on previously unseen data and hence prevents overfitting to the test set.

A total of three different hyperparameters is to be estimated during this phase of training. These include η, μ, and λ which are calculated using a random search technique as proposed by Bergstra & Bengio (2012). This method is reported to be more efficient for hyperparameter optimisation than a traditional grid search. The proposed model contains 159,841 trainable parameters (weights and biases) and is implemented in Python v2.7.13 using the *Keras* v2.0.3 API developed by Chollet et al. (2015). The Keras API is a high-level neural networks API focussing on enabling fast experimentation. The matrix analysis was handled using *Theano* v0.9.0 with GPU support.

3.3.2. Model Validation

To assess the uncertainty of the reported performance a 5-fold cross validation scheme is used with the network setting from the previous evaluation. The network is now trained on 4/5 of the recordings in the AFDB and tested on the remaining 1/5. This process is iterated five times until all recordings have featured in the test set. The final performance is now calculated as the average of the five runs. The generalisation capability of the model is assessed by evaluating the performance of the model on the 48 short-term recordings from the MITDB. This dataset contains 8 AF subjects, but also events of ventricular bigeminy, ventricular trigeminy, atrial flutter and other arrhythmias, which are likely to confound with the learned features of the model (Colloca et al., 2013). Finally, the model is evaluated on the 18 long-term recordings from the NSRDB, which contain no significant arrhythmias. The NSRDB will provide a good estimate of the expected number of FPs in healthy subjects.

4. Results

An exhaustive random sampling approach was performed to obtain the best possible setting of hyperparameters on the training set. A total of three successive trials of 100 parameter settings was conducted, each with reduced search range of the parameters. The final setting is reported in Table 2. Additionally, dropout was applied to the input and recurrent states of the LSTM layer with dropout probability $p = 0.2$.

Table 2: Optimal setting for hyper-parameters on the training set

Parameter	Value
Learning rate, η	0.0013
Momentum coefficient, μ	0.99
Regularization term, λ	0.000 017

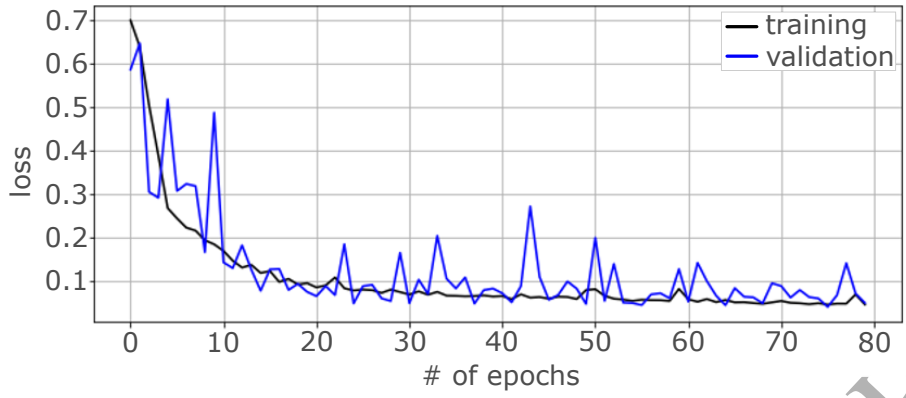


Figure 7: Training (black line) and validation (blue line) cost at each epoch during training.

Training of the network with the parameter settings in Table 2 achieved convergence after approximately 50 epochs. One epoch of training corresponds to the model having iterated over each training example once in the SGD scheme. The training phase of the model is illustrated in Fig. 7, which shows the training and validation loss of the model for each training epoch. The curves show no significant signs of overfitting as the validation curve follows that of the training. The fluctuations in the validation curve is likely due to the relatively small size of the validation set.

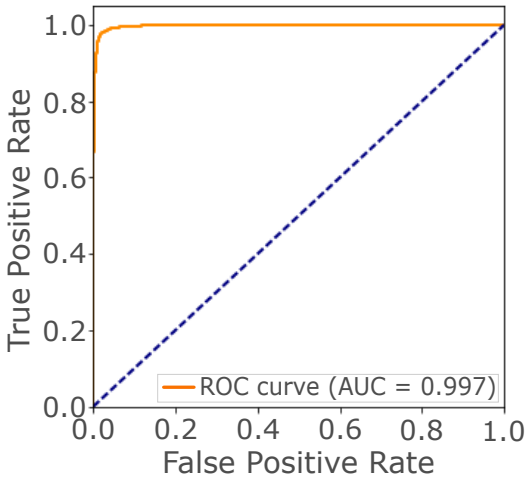
To quantify the effect of the postprocessing scheme, the results are reported both before and after postprocessing. The uncertainties of the results are addressed using a 5-fold crossvalidation scheme. The model is trained for 50 epochs in each of the five folds and the results are listed in Table 3 and the corresponding ROC is illustrated in Fig. 8a. Table 3 clearly shows the improvements of the postprocessed results compared to original results.

Table 3: Results of 5-fold crossvalidation of AFDB

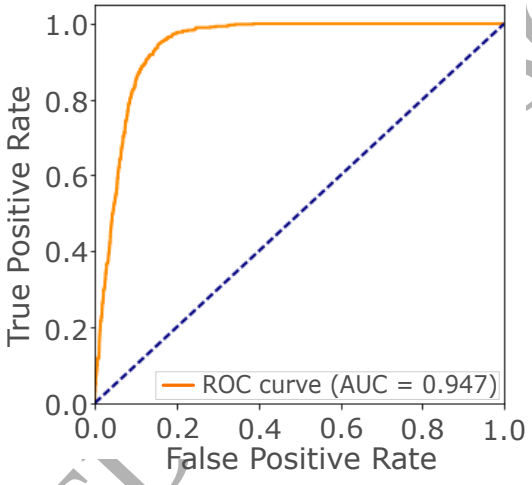
Measure	Original Results (%)	Postprocessed Results (%)
Se	98.17 ± 0.98	98.98 ± 0.21
Sp	96.29 ± 1.61	96.95 ± 1.58
Acc	97.10 ± 0.66	97.80 ± 0.61
PPV	94.99 ± 2.75	95.76 ± 2.67
FPR	3.71 ± 1.77	3.05 ± 1.72

The trained model is validated on the MITDB, which contains 48 short-term recordings which are completely unknown to the model. The MITDB contains many different arrhythmias, but as the model is trained to identify AF, the dataset is converted into a binary classification problem; AF or non-AF. The performance of the model is listed in Table 4 and the corresponding ROC curve is illustrated in Fig. 8b.

Lastly, the model is validated on the NSRDB, which contains 18 long-term recordings without any significant arrhythmias. Most of the performance measures are not applicable is this case because the recordings do not include



(a)



(b)

Figure 8: ROC curves (solid orange lines) together with the curves for random guessing (dashed blue lines). (a) AFDB training set. (b) MITDB test set.

any AF episodes. Validation of the model on the NSRDB provides a good estimate of the expected FPR in healthy subjects.Results reported in Table 4 confirms the capability of the proposed algorithm in detecting AF on unseen datasets.

5. Discussion

The performance achieved on the AFDB beats state-of-the-art results indicating that the unique data-driven features created by the DL model outperforms traditional feature engineering. On the MITDB, the model struggles with a high number of FPs which significantly reduces the performance.

The proposed model includes a total of 159,841 trainable parameters, and hence the computational complexity of this model far exceeds traditional feature engineered methods (Asgari et al., 2015; Dash et al., 2009; Lee et al.,

ACCEPTED MANUSCRIPT

Table 4: Results of the proposed model on two previously unseen data sets

Measure	MITDB (%)	NSRDB (%)
Se	98.96	–
Sp	86.04	95.01
Acc	87.40	–
PPV	45.45	–
FPR	13.96	4.99

2013; Pürerfellner et al., 2014). The model was trained on a NVIDIA GeForce 940M GPU with 384 Cuda cores and 2GB DDR3 memory for 40 minutes before convergence and can analyse and classify 24 hours of RRIs in 0.92 seconds. This computational assessment is performed using a batch size of 256, which is only limited by the memory of the GPU. Any GPU with more dedicated memory will hence speed up the process and allow for even faster classification of long-term recordings.

5.1. Data

For any data-driven model and especially with DL models, it is worth analysing the data. As the features are learned directly from the data and not from prior domain knowledge, as with feature engineering, intuitively the features rely heavily on the data. This model is trained on the AFDB, which contains a good representation of both AF and NSR but also episodes of atrial flutter. The model is trained as a binary classifier, and hence the learned features provide excellent separability between AF and everything else.

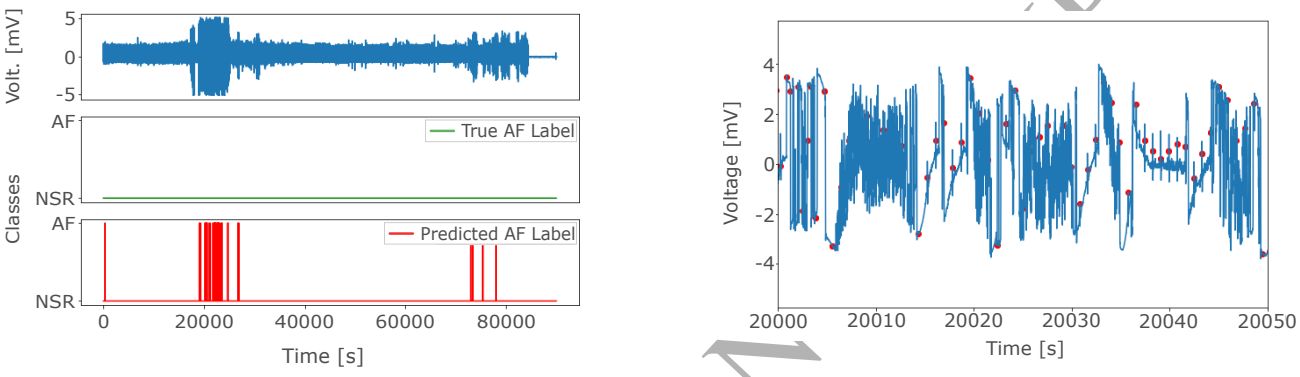
When the model is evaluated on the MITDB, the class distribution of the data have suddenly changed. New and previously unseen types of arrhythmias such as ventricular bigeminy and trigeminy may confound with the learned features of the model (Colloca et al., 2013). The sudden introduction of new arrhythmias could explain the significant increase in FPs when comparing to the results from the AFDB in Table 3. To investigate this, a modified evaluation is performed. Table 5 presents the results achieved on the MITDB in which segments with certain arrhythmias are excluded. For instance, the results reported in the third column (Paced Beats) show the performance of the algorithm on the MITDB by excluding the RRIs for the Paced Beats. Similarly, the fourth column (V. Bigeminy & Trigeminy) report the results by excluding theses type of beats from the RRIs. It should be mentioned that the reported results in Table 5 have all been post processed.

The results of this analysis suggest that neither paced beats, ventricular bigeminy or trigeminy had a significant impact on the number of FPs. For example, By excluding the subject with paced beats and neglecting any FPs within episodes of ventricular bigeminy and trigeminy, the FPR was reduced from 13.96% to 12.06%. These results indicate that the investigated arrhythmias are not alone responsible for the high number of FPs. This study only investigates the classification of AF from NSR. However, the proposed model can be further developed to distinguish different arrhythmia using a multi-class classification approach in order to further reduce the FPs.

The model was also evaluated on the NSRDB which contains no significant arrhythmias and hence provides a

Table 5: Modified results obtained on the MITDB. Segments with certain arrhythmias have been excluded from the dataset. Each column indicates which arrhythmias have been excluded. All results have been postprocessed.

Measure	Original Results (%)	Paced Beats (%)	V. Bigeminy & Trigeminy (%)	Combined (%)
Se	98.96	99.82	98.96	99.82
Sp	86.04	84.87	88.91	87.94
Acc	87.40	86.53	90.00	89.30
PPV	45.45	45.18	52.09	51.71
FPR	13.96	15.13	11.09	12.06



(a) (b)

Figure 9: False predictions during NSR. Noisy ECG signals prevent detection of R-peaks and introduce artifacts in the RRI time-series resulting in segments wrongly classified as AF. (a) ECG signal along with true and predicted labels, (b) False predictions during NSR. Noisy ECG signals prevent detection of R-peaks and introduce artifacts in the RRI time-series resulting in segments wrongly classified as AF.

good estimate of the expected number of FPs in healthy subjects. Comparing the results obtained on the NSRDB to the results on the AFDB shows an increase in the FPR from 3.05% to 4.99%, which suggest further investigation of the data in the NSRDB. Inspecting the raw ECG signals reveals multiple noisy segments in which the automatic R-peak segmentation algorithm has failed to detect the R-peaks. This translates directly into artifacts in the RRI times-series as the ventricular rhythm suddenly appears irregular, resulting in segments being wrongly classified as AF as illustrated in Fig. 9.

R-peak segmentation in noisy ECG signals requires clever filtering more advanced algorithms than was used for annotating the NSRDB. An alternative solution would be to classify the noise level in the ECG signal for each segment and reject segments surpassing an estimated threshold. In this way, segments containing a high noise level would be rejected instead of wrongly classified as AF.

5.2. Postprocessing

The postprocessing scheme applied on this algorithm is a quick and efficient solution to correct single segment artifacts or outlier. The idea is to nullify any such occurrence because AF is typically seen in episodes and not in quick bursts. By visualising the predictions on entire recordings along with the corresponding true rhythm

annotation, it is clear that the postprocessing scheme enhances the performance. In Fig. 10, the predicted labels of two recordings from the test set of the AFDB is illustrated with corresponding true labels before and after the postprocessing. Notice how single segment misclassifications are corrected and aligned with the true label. The width of the median filter could be further increased to strengthen this effect but at the cost of missing short episodes of paroxysmal AF.

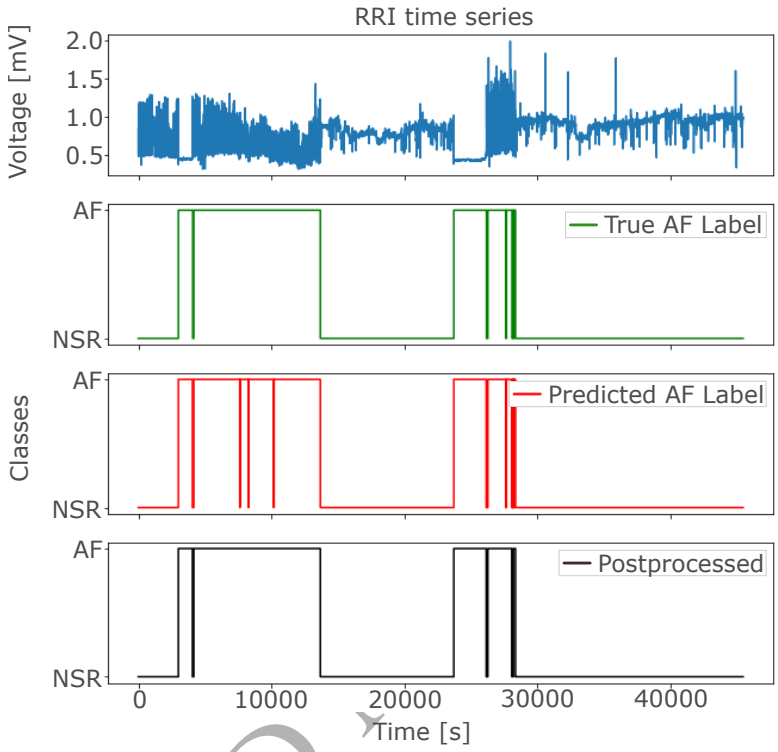


Figure 10: Effect of post-processing scheme: top figure illustrates the raw RRIs of record 08378 in the AFDB. The three remaining figures show the true AF label (green) along with the predicted AF label (red) and the predicted label after post-processing (black). Notice how the single segment misclassifications are corrected after the post-processing scheme is applied.

In this study, the entire recording is processed at once, and hence it is possible to utilise the predictions of both past and future inputs in the postprocessing scheme. Because the processing is non-causal, it is not applicable in a real-time setting. Even though the filtering method is not suitable, the general idea of smoothing the output predictions is still viable. A real-time implementation of this system would require a buffer which continuously kept track of previous predictions. This is easily implemented but less efficient if the entire record is already available as in this study.

6. Conclusion

In this work, an automatic real-time AF detection method based on the DL approach was proposed. It eliminates the need for traditional feature engineering, as the DL model learns data-driven features to distinguish AF from the remaining rhythms in the signals. Therefore, this algorithm can be easily used “in house” by clinicians/cardiologists

ACCEPTED MANUSCRIPT

to detect AF. The proposed method shows a high performance on the AFDB dataset with a Se and Sp of 98.98% and 96.95%, respectively. Furthermore, the proposed post-processing scheme successfully reduced the number of FPs, which resulted in a decrease in the FPR from 3.71% to 3.05%.

Further validation of the proposed model on the MITDB revealed a significantly higher number of FPs and hence a reduction in Sp to 87.70% and an increased FPR of 12.21%. These results suggested that the previously unseen rhythms in the MITDB might confound with the learned features of the model. Because the features were learned directly from the data, it was impossible to account for any rhythms not contained in the training set and hence it was expected to see an increased FPs. The model was also tested on the NSRDB which contains 18 long-term recordings with no significant arrhythmias. The test provided a good estimate of the expected number of FPs in a healthy subject. The results indicated a relatively high FPR of 4.99%. Further investigation of the NSRDB confirmed that the automatic R-peak segmentation algorithm used for the beat annotations had failed to detect R-peaks in multiple segments of the recordings due to noisy ECG signals. The mis-classified R-peaks introduced artifacts in the RRI time-series in which the ventricular rhythm appeared irregular. Inspection of the model's performance on the noisy ECG segments indicated a high number of FPs as expected. A possible solution to this problem would be to identify the noise level in each ECG segment and reject segments with a noise level higher than a predefined threshold.

Acknowledgements

The authors would like to thank Dr. Erik S. Poulsen from Cortrium ApS for his constructive comments and suggestions during the project period.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

References

Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A., & San Tan, R. (2017). A deep convolutional neural network model to classify heartbeats. *Computers in biology and medicine*, *89*, 389–396.

Al Rahhal, M. M., Bazi, Y., AlHichri, H., Alajlan, N., Melgani, F., & Yager, R. R. (2016). Deep learning approach for active classification of electrocardiogram signals. *Information Sciences*, *345*, 340–354.

Alonso-Atienza, F., Rojo-Alvarez, J. L., Rosado-Munoz, A., Vinagre, J. J., Garcia-Alberola, A., & Camps-Valls, G. (2012). Feature selection using support vector machines and bootstrap methods for ventricular fibrillation detection. *Expert Systems with Applications*, *39*, 1956–1967.

Asgari, S., Mehrnia, A., & Moussavi, M. (2015). Automatic detection of atrial fibrillation using stationary wavelet transform and support vector machine. *Computers in biology and medicine*, *60*, 132–142.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Ceylan, R., zbay, Y., & Karlik, B. (2009). A novel approach for classification of ECG arrhythmias: Type-2 fuzzy clustering neural network. *Expert Systems with Applications*, *36*, 6721 – 6726.

Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv e-prints*, . arXiv:1412.3555.

Colloca, R., Johnson, A. E., Mainardi, L., & Clifford, G. D. (2013). A support vector machine approach for reliable detection of atrial fibrillation events. In *Computing in Cardiology Conference (CinC), 2013* (pp. 1047–1050). IEEE.

Dash, S., Chon, K., Lu, S., & Raeder, E. (2009). Automatic real time detection of atrial fibrillation. *Annals of biomedical engineering*, *37*, 1701–1709.

Fuster, V., Rydén, L. E., Cannom, D. S., Crijns, H. J., Curtis, A. B., Ellenbogen, K. A., Halperin, J. L., Le Heuzey, J.-Y., Kay, G. N., Lowe, J. E. et al. (2006). Acc/aha/esc 2006 guidelines for the management of patients with atrial fibrillation. *Circulation*, *114*, e257–e354.

Geras, K. J., Mohamed, A.-r., Caruana, R., Urban, G., Wang, S., Aslan, O., Philipose, M., Richardson, M., & Sutton, C. (2015). Blending lstms into cnns. *ArXiv e-prints*, . arXiv:1511.06433.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, *101*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

ACCEPTED MANUSCRIPT

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning* (pp. 1764–1772).

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, *18*, 602 – 610. IJCNN 2005.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*, 1735–1780.

Homaeinezhad, M., Atyabi, S., Tavakkoli, E., Toosi, H., Ghaffari, A., & Ebrahimpour, R. (2012). Ecg arrhythmia recognition via a neuro-svmknn hybrid classifier with virtual QRS image-based geometrical features. *Expert Systems with Applications*, *39*, 2047 – 2058.

Khalaf, A. F., Owis, M. I., & Yassine, I. A. (2015). A novel technique for cardiac arrhythmia classification using spectral correlation and support vector machines. *Expert Systems with Applications*, *42*, 8361 – 8368.

Khorrami, H., & Moavenian, M. (2010). A comparative study of dwt, cwt and dct transformations in ECG arrhythmias classification. *Expert Systems with Applications*, *37*, 5751 – 5757.

Kiranyaz, S., Ince, T., & Gabbouj, M. (2016). Real-time patient-specific ecg classification by 1-D convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, *63*, 664–675.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lake, D. E., & Moorman, J. R. (2010). Accurate estimation of entropy in very short physiological time series: the problem of atrial fibrillation detection in implanted ventricular devices. *American Journal of Physiology-Heart and Circulatory Physiology*, *300*, H319–H325.

Le Heuzey, J.-Y., Paziaud, O., Piot, O., Said, M. A., Copie, X., Lavergne, T., & Guize, L. (2004). Cost of care distribution in atrial fibrillation patients: the cocaf study. *American heart journal*, *147*, 121–126.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*, 436.

Lee, J., Reyes, B. A., McManus, D. D., Maitas, O., & Chon, K. H. (2013). Atrial fibrillation detection using an iphone 4s. *IEEE Transactions on Biomedical Engineering*, *60*, 203–206.

Majumdar, A., & Ward, R. (2017). Robust greedy deep dictionary learning for ecg arrhythmia classification. In *Neural Networks (IJCNN), 2017 International Joint Conference on* (pp. 4400–4407). IEEE.

ACCEPTED MANUSCRIPT

Martis, R. J., Acharya, U. R., Mandana, K., Ray, A., & Chakraborty, C. (2012). Application of principal component analysis to ECG signals for automated diagnosis of cardiac health. *Expert Systems with Applications*, *39*, 11792 – 11800.

Mjahad, A., Rosado-Muoz, A., Bataller-Mompen, M., Francs-Vllora, J., & Guerrero-Martnez, J. (2017). Ventricular fibrillation and tachycardia detection from surface ecg using time-frequency representation images as input dataset for machine learning. *Computer Methods and Programs in Biomedicine*, *141*, 119 – 127.

Moavenian, M., & Khorrami, H. (2010). A qualitative comparison of artificial neural networks and support vector machines in ECG arrhythmias classification. *Expert Systems with Applications*, *37*, 3088 – 3093.

Moody, G. B., & Mark, R. G. (1983). A new method for detecting atrial fibrillation using rr intervals. *Comput. Cardiol.*, (pp. 227–230).

Moody, G. B., & Mark, R. G. (2001). The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, *20*, 45–50.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O(1/k2). *Sov. Math. Dokl.*, *27*, 372–376.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Pawiak, P. (2018). Novel methodology of cardiac health recognition based on ECG signals and evolutionary-neural system. *Expert Systems with Applications*, *92*, 334 – 349.

Pürerfellner, H., Pokushalov, E., Sarkar, S., Koehler, J., Zhou, R., Urban, L., & Hindricks, G. (2014). P-wave evidence as a method for improving algorithm to detect atrial fibrillation in insertable cardiac monitors. *Heart Rhythm*, *11*, 1575–1583.

Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., & Ng, A. Y. (2017). Cardiologist-level arrhythmia detection with convolutional neural networks. *ArXiv e-prints*, . arXiv:1707.01836.

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on* (pp. 4580–4584). IEEE.

Shadmand, S., & Mashoufi, B. (2016). A new personalized ECG signal classification algorithm using block-based neural network and particle swarm optimization. *Biomedical Signal Processing and Control*, *25*, 12 – 23.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *ArXiv e-prints*, . arXiv:1506.04214.

Stein, P. K., Bosner, M. S., Kleiger, R. E., & Conger, B. M. (1994). Heart rate variability: a measure of cardiac autonomic tone. *American heart journal*, *127*, 1376–1381.

ACCEPTED MANUSCRIPT

Stewart, S., Murphy, N., Walker, A., McGuire, A., & McMurray, J. (2004). Cost of an emerging epidemic: an economic analysis of atrial fibrillation in the uk. *Heart*, *90*, 286–292.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139–1147).

Wu, Z., Ding, X., Zhang, G., Xu, X., Wang, X., Tao, Y., & Ju, C. (2016). A novel features learning method for ECG arrhythmias using deep belief networks. In *Digital Home (ICDH), 2016 6th International Conference on* (pp. 192–196). IEEE.

Yu, S.-N., & Chou, K.-T. (2008). Integration of independent component analysis and neural networks for ECG beat classification. *Expert Systems with Applications*, *34*, 2841–2846.

Zoni-Berisso, M., Lercari, F., Carazza, T., & Domenicucci, S. (2014). Epidemiology of atrial fibrillation: European perspective. *Clin. Epidemiol.*, *6*, 213–20.

Zubair, M., Kim, J., & Yoon, C. (2016). An automated ecg beat classification system using convolutional neural networks. In *IT Convergence and Security (ICITCS), 2016 6th International Conference on* (pp. 1–5). IEEE.

ACCEPTED MANUSCRIPT