Noise Cleaning of ECG on Edge Device Using Convolutional Sparse Contractive Autoencoder

Rohan Banerjee
TCS Research
Tata Consultancy Services
rohan.banerjee@tcs.com

Ayan Mukherjee
TCS Research
Tata Consultancy Services
ayan.m4@tcs.com

Avik Ghose
TCS Research
Tata Consultancy Services
avik.ghose@tcs.com

2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 978-1-6654-1647-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/PerComWorkshops53856.2022.9767313

***Abstract*—Single-lead Electrocardiogram (ECG) can be easily measured by a commercial smartwatch or a dedicated wearable device. The waveforms are often susceptible to background noise and motion artifacts introducing errors in disease interpretation. An effective yet light-weight de-noising of ECG is an open area of research. In this paper, we propose a novel convolutional autoencoder structure considering a number of regularization terms like sparsity constraint, contractive regularization and L2 norm for ECG de-noising. The deep learning model is duly optimized to efficiently run on low-power edge devices. The proposed approach is evaluated on a simulated and a real-world single-lead ECG database recorded from normal subjects as well as patients having Atrial Fibrillation (AF) and other kinds of abnormal heart rhythms. A thorough comparison is performed with a number of related signal processing and deep learning based prior approaches. Experimental results show that the proposed autoencoder yields the least Root Mean Square Error (RMSE) in reconstruction of clean signals from input ECG corrupted due to addition of noise. Our approach is also able to preserve the relevant morphological properties in the reconstructed ECG data for successful detection of AF and other abnormal rhythms. The optimized model is deployed on a low-power single-board computer for real-time noise cleaning.**

***Index Terms*—Electrocardiogram, Autoencoder, Convolutional Neural Network, Noise cleaning, TinyML**

I. Introduction

Electrocardiogram (ECG) measures the electrical activities of the heart. Although 12-lead ECG devices are installed in clinics, a cheaper and more unobtrusive solution is the single-lead ECG popularly used for community screening. There are commercial wearable devices available in the form factor of a smartwatch or a wrist-band capable of recording single-lead ECG for personal health monitoring and on-device fitness tracking. An automatic diagnosis from the digitally recorded ECG is possible thanks to the advancement in artificial intelligence, machine learning and deep learning techniques. Apart from measuring heart rate, such solutions can detect the presence of cardiac anomalies like arrhythmias or Atrial Fibrillation (AF). An ECG signal contains three major components, the P wave, the QRS complex and the T wave in each cycle. Owing to the complex nature, ECG signals are prone to background noise. The high frequency noise components may occur due to the misplace of sensor electrodes on human body during recording, electromagnetic induction of nearby devices and motion artifacts. There can also be a low frequency baseline drift due to breathing. Presence of noise in ECG may heavily impact the performance of the automated diagnosis algorithms. The eHealth systems thus require an accurate reconstruction of clean data from the contaminated recordings as a basic pre-processing step at the device front-end.

The wide-band background noise often overlaps the fundamental frequency regions in ECG. A bandpass filter may not remove the in-band noise components, requiring a more complex noise cleaning treatment. Adaptive filter based non-linear approach was used in [1] for noise cleaning of ECG. Independent Component Analysis (ICA) [2], Empirical Mode Decomposition (EMD) [3] or wavelet based techniques were also explored. Deep learning approaches have been very popular in recent days. A multi-layer Convolutional Neural Network (CNN) regressor was proposed by Arsene *et al.* [4] for ECG noise cleaning. Chiang *et al.* [5] proposed an architecture of convolutional de-noising autoencoder. Both approaches were evaluated on the public MIT-BIH Arrhythmia database. Although the modern deep learning based noise cleaning approaches outperform the traditional signal processing based approaches, they are still not reliable in complete reconstruction of data in presence of high amplitude noise. Most of them are evaluated on 12-lead ECG, recorded using medical-grade equipments in a controlled environment (hospital ICU) under restricted patient movements. However, single-lead ECG sensors are of inferior quality than clinical devices and are typically used in real life applications, e.g. real-time heart rate monitoring on a smartwatch during exercise. Hence, the signals are in general noisier than clinical recordings. The available approaches mostly evaluate their noise cleaning performance in terms of quantitative error measuring parameters like mean squared error. Usability of the reconstructed data in real world applications were not properly evaluated in literature. A typical deep learning approach results in a large model which is resource hungry and typically needs a powerful computer or an external server to run. In recent times, attention on running complex deep learning architectures on smaller edge devices is getting more attention due to latency and lower power consumption. They are particularly important in heathcare applications to maintain user privacy, as the model entirely runs on smaller personal devices without sending the data to a server. In this paper, we propose an autoencoder architecture designed to run on edge devices for real-time ECG de-noising. Major contributions of this paper are:

- A novel convolutional autoencoder architecture is defined for noise cleaning of single-lead ECG signals.
- A thorough evaluation is done on a simulated dataset and a real open-access database, recorded using commercially available single-lead device and our approach is compared with a number of existing approaches.
- Apart from measuring the goodness of signal reconstruction, a qualitative study is performed to test the usability of the reconstructed data by evaluating on a benchmark cardiac rhythm detection algorithm.
- The deep learning model is optimized to reduce its size to run on low-power edge devices without compromising the performance. A prototype system is implemented for real-time noise cleaning.

Rest of the paper is organized as follows. Section II provides a detailed description of the proposed autoencoder structure. Model optimization techniques and experimental results are discussed in Section III and IV followed by a conclusion.

II. Proposed De-noising Autoencoder Structure

An autoencoder is a neural network, popularly used for data encoding in an unsupervised manner [6]. It has two major components, the encoder and the decoder. The encoder compresses the input to an encoded representation of reduced dimension, called the latent vector. The decoder tries to reconstruct the original input from the latent vector. Both encoder and decoder are designed using non-linear neural networks. The difference between the original input and the reconstructed output is called the reconstruction loss. During training, the autoencoder tries to minimize the reconstruction loss using back propagation. Hence, the latent vector becomes an efficient representation of the input in a reduced space. Autoencoders are popularly used in data compression, anomaly detection and signal or image de-noising. Basic block diagram of a

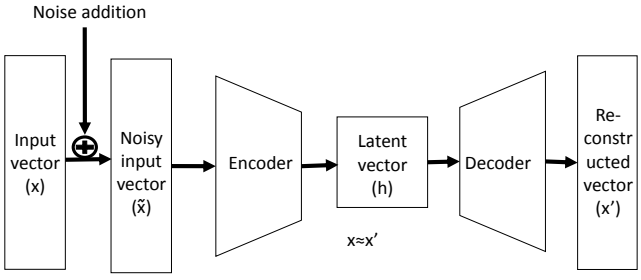


Fig. 1. Structure of a basic de-noising autoencoder with encoder and decoder

de-noising autoencoder for 1D signals is shown in Fig. 1. Considering the simplest structure of having a single hidden layer in the encoder and the decoder, the autoencoder first takes the original non-noisy input, $x \in \mathbb{R}^n = X$ and corrupts it as $\tilde{x} \in \mathbb{R}^n = \tilde{\mathcal{X}}$ by some stochastic mapping, $\tilde{x} \sim q(\tilde{x}|x)$. Subsequently, the encoder takes the noisy data, $\tilde{\mathcal{X}}$ as input and maps it to the latent representation, $h \in \mathbb{R}^s = \mathcal{F}$. The decoder reconstructs the original non-noisy input as $x' \in \mathbb{R}^n = \mathcal{X}'$. The encoder and the decoder employ the non-linear functions σ and σ' for the transformations.

$$h = \sigma(W\tilde{x} + b) \quad (1)$$

$$x' = \sigma'(W'h + b') \quad (2)$$

Where, W, b, W' and b' are the trainable weight and bias terms of the encoder and the decoder. The autoencoder is trained end to end to minimize the mean squared error between the original non-noisy input and the reconstructed data as loss function, which is given by:

$$\mathcal{L}(x, x') = \frac{1}{m}\sum_{i=1}^{m} ||x^{(i)} - x'^{(i)}||^2 \quad (3)$$

Here, m denotes the number of training instances. Due to non-linear activation, a complex neural network with a large number of neurons is prone to overfit. We apply a number of regularizers to the autoencoder for an efficient learning of the training data. In order to control the overall activation of the encoder, a sparsity penalty term is added which limits the number of active neurons in the encoder. If $h_j(x^{(i)})$ is the activation of j^{th} neuron for input $x^{(i)}$, then $\hat{\rho_j}$, the average activation of that neuron for all the training instances is:

$$\hat{\rho_j} = \frac{1}{m}\sum_{i=1}^{m} h_j(x^{(i)}) \quad (4)$$

During training, $\hat{\rho_j}$ is forced to be close to a predefined parameter, ρ of very small value (0.03 in our case) by minimizing the Kullback-Leibler (KL) divergence between them as a penalty term, which is added to the loss function.

$$KL(\rho||\hat{\rho_j}) = \rho \cdot log\frac{\rho}{\hat{\rho_j}} + (1-\rho) \cdot log\frac{1-\rho}{1-\hat{\rho_j}} \quad (5)$$

This sparsity constraint regularizes the activation of the neurons in the encoded layer and forces the training model to respond to the unique statistical properties of the training data. We add another regularizer to force the learning model to be less sensitive to the minor fluctuations of the training data and more sensitive to the important variations. This corresponds to the Frobenius norm of the Jacobian matrix ($J_f(x)$) of the encoder activation with respect to the input. This is termed as the contractive regularization [7], which forces the model to learn useful information about the training distribution. The regularizer is calculated as the sum of squares of the partial derivatives of the encoded vector with respect to input, as:

$$||J_f(x)||_F^2 = \sum_{i=1}^{m} ||J_f(x^{(i)})||_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{s}\sum_{k=1}^{n}\left(\frac{\partial h_j(x^{(i)})}{\partial x_k^{(i)}}\right)^2 \quad (6)$$

Finally, the neuron weights of the encoder and decoder are regularized using L2 norm. Objective function of the proposed sparse contractive autoencoder from eqn. 3, 5 and 6 becomes:

$$\begin{aligned} J = \mathcal{L}(x, x') + \beta.\sum_{j=1}^{s} KL(\rho||\hat{\rho_j}) + \lambda_1.||J_f(x)||_F^2 \\ + \lambda_2.\sum_{k=1}^{n}\sum_{j=1}^{s}\left(W_{jk}^2 + {W'_{kj}}^2\right) \end{aligned} \quad (7)$$

In eqn. 6, n and s denote input length and number of neurons in the encoder layer; β, λ_1 and λ_2 are hyper-parameters, controlling the strength of different regularizers.

In a convolutional autoencoder, CNN is used as a basic building block for defining the encoder and the decoder. CNN

is a powerful deep learning architecture that can capture the spatial and temporal dependencies in an input via convolution operation through a set of filters (kernel) to create a relevant feature map. In a multi-layer CNN, the first few convolutional layers capture the low-level features, whereas the denser layers are responsible for capturing more complex high-level features. Although CNNs are widely applied in image processing related applications, 1D CNNs have been popular in analysis of time series data like ECG. Architecture of our proposed 1D convolutional de-noising autoencoder is shown in Fig. 2. Tensor dimension at the output of each layer is duly mentioned in the block diagram. Noisy ECG signals are broken into non-overlapping rectangular windows of 1024 samples. Each window is considered as independent input to the autoencoder. The encoder contains a total of 3 convolutional layers. The kernel size is selected as 5 for performing the convolution operation and 20 filters are used in each layer. Zero padding is applied to the inputs in each convolutional layer for retaining the original dimension. Rectified Linear Unit (ReLU) function is used for non-linear activation, where $ReLU(x) = max\{0, x\}$. Dimension of the output feature map in each convolutional layer is reduced by the associated 1D maxpool layer, having a pool size of 2. The feature map at the end of the third maxpool layer is flattened and applied to a dense layer to convert to the encoded latent vector of length 512. The decoder follows an inverse structure to the encoder. However, the maxpool layers are replaced by upsample layers to gradually increase the dimension of the output feature map. There is an additional convolutional layer in the decoder, having a single filter with kernel size of 5 followed by a sigmoid activation function to convert the reconstructed data to the original 1D data space of the input. Sparsity and contractive regularizations are only applied to the convolutional and dense layers of the encoder, whereas L2 norm is applied to all but the final convolutional layer of the decoder.

Implementation of the proposed network is done in Python

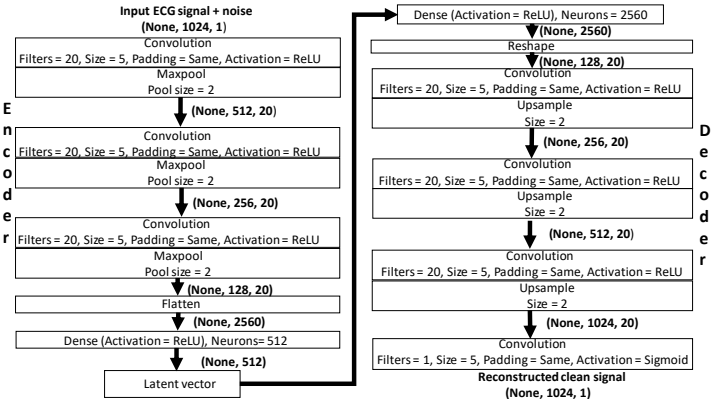


Fig. 2. Architecture of the proposed convolutional de-noising autoencoder

3.7.3 using TensorFlow 2.5.0. The hyper-parameters are tuned based on random search. During training, the objective function is minimized using an Adam optimizer with learning rate of 0.001, 300 epochs and mini batch size of 64. The Sparsity penalty term (β) is set to 1, λ_1 and λ_2 are set to 0.01 and 0.03. Initial weights of the neurons in the convolutional and the dense layers are set using Xavier initialization [8]. Here, initial weights of a layer are randomly assigned from a Gaussian distribution of zero mean and variance, $var = \frac{2}{n_{in}+n_{out}}$, n_{in} and n_{out} are the number of input and output neurons. The bias terms are initialized by zeros.

III. Model Optimization for Edge Devices

The baseline model of the proposed autoencoder is trained on a desktop having Intel® Core i7-8550U, 4.0 GHz processor, 16 GB of memory and an Nvidia graphics processing unit. A standard deep learning model trained on a desktop or a server needs significant optimization in order to efficiently run on low-power edge devices. In our application, we select Raspberry Pi 3, Model B+ as the target edge device to run the ECG de-noising application in real-time. The following optimization steps are applied to compress the baseline model.

A. Depthwise Separable Convolution

The standard convolution operations in the baseline autoencoder are replaced by depthwise separable convolution. In this process, each convolutional layer first performs a depthwise convolution task that acts separately on the channels, followed by a pointwise convolution that mixes the channels. Although the process is similar to the standard convolution operation on single channel input, it generates much fewer parameters from the second convolutional layer of the proposed autoencoder structure having multiple input channels because of the number of filters applied to the previous convolutional layer. This process not only reduces the model size due to fewer parameters but also results in a faster inference speed on target device due to fewer mathematical operations.

B. Weight Pruning

Model size can be significantly reduced by eliminating the lesser important connections via replacing them with zeros. A significant number of connections in a neural network model is found close to zero. Weight pruning is a technique to gradually zero out few such insignificant weights during training in order to achieve model sparsity. Such pruned models are easy to compress which reduces the model storage area without a significant impact on model accuracy. Weight pruning requires a retraining of the baseline model via gradually increasing sparsity in an iterative manner. The baseline model weights are taken as initial weights. We start by adding 30% of sparsity to the baseline model and finish at 50% of sparsity using a polynomial decay function causing a minimum performance drop in the pruned model. The perturbed model is retrained in the process to minimize the loss function. The retraining is done for 50 epochs at a lesser learning rate (0.00005) compared to the baseline model.

C. Post Training Quantization

A model trained on a high end computer system may not run effectively on a low-power edge device due to hardware

constraints. In post training quantization, the pruned model is further optimized by quantizing the 32 bit floating point weights of the trained model into 8 bit integers. This process not only reduces the model size but also improves CPU and hardware accelerator latency, with a little impact on model performance. We apply dynamic range quantization on the pruned model which statistically quantizes the weights. During inferencing, the weights are reconverted from 8-bits of precision to floating points using floating-point kernels. The conversion is done only once at the target device and cached to reduce latency.

TensorFlow Lite APIs are used to optimize the baseline model for running on the target device. The proposed optimization process causes 8 times reduction in the size of the baseline model after compression (the baseline model size is 2.8 MB) along with 10 times faster performance on the target device.

IV. Data Analysis and Experimental Results

A. Dataset Description

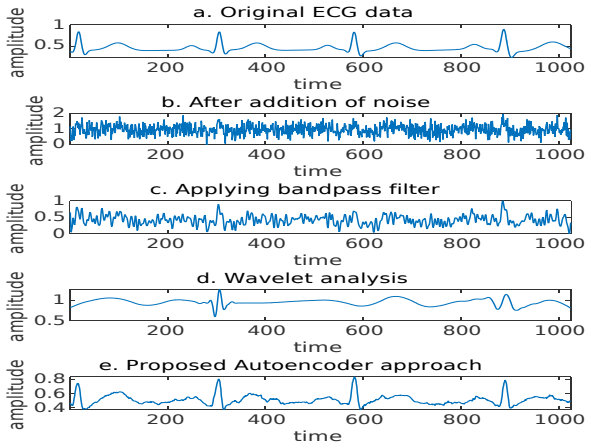
The baseline autoencoder is trained and evaluated on a simulated and a real ECG dataset. The simulated dataset is generated using the ECGSYN software [9]. The user can externally provide mean heart rate and can also introduce morphological variations in the simulated waveforms. A total of 12,000 ECG instances each having 1024 samples are simulated at a sampling rate of 300 Hz. Heart rate in the generated data ranges from 55 bpm to 110 bpm. A total of 8000 recordings are randomly selected for creating the training set and 2000 instances are kept for internal validation in order to determine the network hyper-parameters via trial and error. The remaining 2000 instances are used for test purpose.

A real-world annotated database is further considered for qualitatively evaluating the impact of noise cleaning on the performance of available machine learning based ECG classifiers. The PhysioNet Challenge 2017 database [10] is a large open-access database containing single-lead ECG recorded using commercially available portable devices. Out of 8528 recordings, 5154 recordings are annotated as normal sinus rhythm, 771 are of AF, 2557 are of other types of abnormal rhythms and 46 recordings are marked as noisy and can not be diagnosed. The signals are sampled at 300 Hz and the duration varies from 9 seconds to 61 seconds. Similar to the simulated dataset, recordings of the PhysioNet database are first broken into non-overlapping windows of 1024 samples, resulting in a total of 75,119 instances. A small fraction of data (15%) from the database is added to the previous training set of simulated data to create the final learning model for denoising. The remaining 85% of data is used for noise cleaning and evaluating the usability of the reconstructed data in cardiac rhythm classification using a popular machine learning algorithm. It is ensured that multiple windows obtained from a single recording are not mixed up in the training and test sets. The original signals are corrupted by adding white Gaussian noise. In order to ensure the model is not over-fitted, three different Signal to Noise Ratio (SNR) of 2 db, 0 db and -2 db are assigned to the training and validation sets, whereas SNR of the two test sets is set to 3 db, 0 db and -3 db.

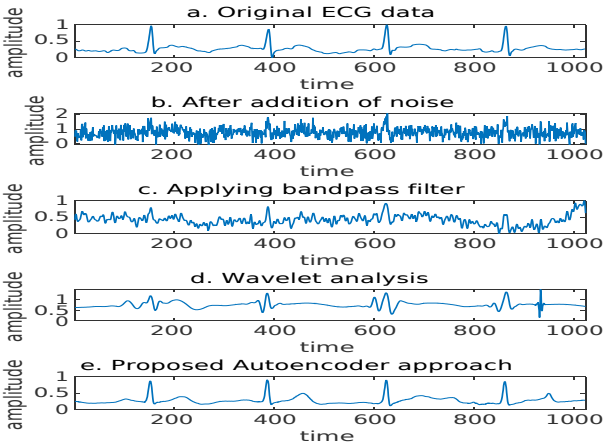
B. Experimental Results of the Baseline Autoencoder

The proposed baseline autoencoder is first compared with two popular signal processing approaches, 1) 4^{th} order butterworth bandpass filter having cut-off frequencies of 0.5 Hz and 40 Hz and 2) Discrete Wavelet Transform (DWT) based noise cleaning. Here, the noisy signals are decomposed upto fifth level using Symlets 7 (sym7) mother wavelet. For each level, adaptively selected soft thresholding is applied to the detail coefficients. The cleaned signals are reconstructed from the fifth order approximate coefficients and the modified detail coefficients. Fig. 3 plots sample waveforms reconstructed from noise contamination (SNR = 0 db) for a comparative analysis. It clearly shows that morphology of the reconstructed signals are the least affected by the proposed autoencoder structure on normal and abnormal ECG data. A detailed quantitative evaluation is shown in Fig. 4. The reconstruction loss is reported in terms of Root Mean Square Error (RMSE) between the original non-noisy and the cleaned waveforms. Apart from filter and wavelet based de-nosing, we consider two more deep learning based prior approaches in [4] and [5] for comparison. Arsene *et al.* [4] proposed a CNN regressor, where the noisy signals pass through a number of convolutional and pooling layers followed by a dense layer that creates the reconstructed signals. Chiang *et al.* [5] proposed a convolutional de-noising autoencoder. It can be observed that our proposed approach as well as the prior deep learning approaches outperform the filter and wavelet based de-noising methods. However, owing to the rigorous regularization steps, the proposed autoencoder efficiently learns the statistical properties of the training data at different SNR levels and is less vulnerable to the minor fluctuation. Hence, it yields the least RMSE in overall signal reconstruction compared to the prior deep learning approaches.

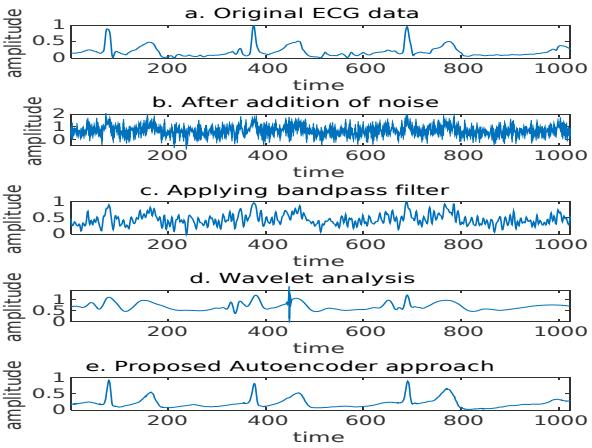
We perform a qualitative evaluation of the baseline autoencoder in terms of usability of the reconstructed signals in a real application. The open source machine learning algorithm in [11] for identifying normal, AF and other abnormal rhythms is selected for this purpose. The algorithm derives a set of 150 hand crafted features, related to ECG morphology and short term heart rate variability. The feature set is applied to a series of cascaded binary Adaptive Boosting classifiers [11]. Performance of the algorithm heaviliy relies on the noise cleaning of ECG in the pre-processing, as features calculated on noisy ECG can be erroneous which causes a neagtive impact on disease detection. Classification performance is reported by the metric, F_t, as provided in the PhysioNet Challenge [10], where $F_t = \frac{F_{norm}+F_{af}+F_{oth}}{3}$ and F_{norm}, F_{af}, F_{oth} represent the F1 score of detecting normal, AF and other abnormal rhythms. As mentioned earlier, 85% of data selected from the PhysioNet database as the test set is used for evaluation. Median value of the metric, F_t is measured as 0.85 on the original non-noisy database in 5 fold cross validation. Subsequently, the signals are contaminated by adding noise and are broken into instances of 1024 samples. The signals



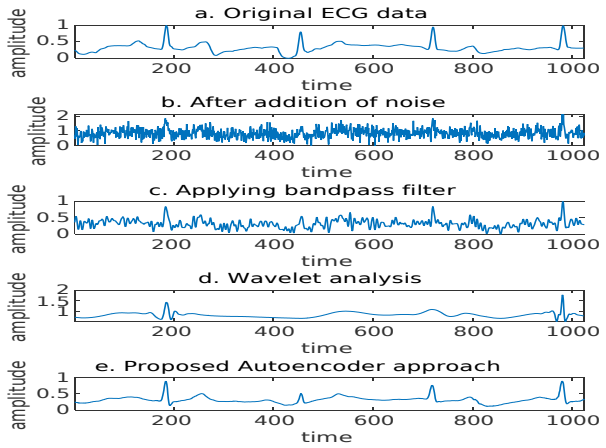
(a) Sample recording from simulated dataset



(b) Normal recording from PhysioNet database

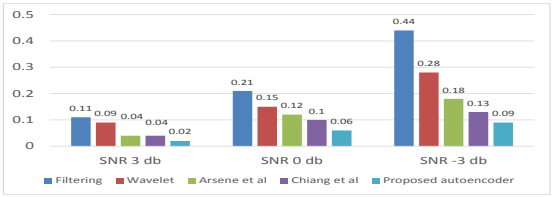


(c) AF recording from PhysioNet database

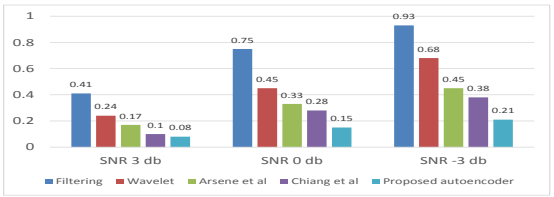


(d) Non-AF abnormal recording from PhysioNet database

Fig. 3. Performance of different noise cleaning techniques on sample recordings. In all cases, SNR of original signal is set to 0 db by addition of noise



(a) RMSE values on simulated test dataset



(b) RMSE values on PhysioNet test database

Fig. 4. Performance comparison with existing approaches in RMSE

are de-noised and merged according to their original length.

Finally, the disease classification performance is measured on the reconstructed signals. Table I shows the impact of various de-noising approaches on classification performance. Although the existing approaches are effective at an SNR of 3 db where the signals are less corrupted, their performance tends to drop with the reduction in SNR (there is significant performance drop in disease detection rate at SNR = -3 db). Whereas, our proposed approach is more reliable to reconstruct the signal morphology at all noise levels. The disease specific features are thus less erroneous, resulting in a commending classification performance even at an SNR of -3 db.

C. Prototype System for Real-time Noise Cleaning

Table II shows a comparative performance between the optimized smaller autoencoder model and the baseline model in terms of average reconstruction loss at various SNR levels considered in the paper. The optimized model runs on a Raspberry Pi 3, Model B+ during the process. It can be observed that there is no significant performance drop in overall RMSE

TABLE I
COMPARISON AMONG NOISE CLEANING APPROACHES IN DETECTION OF VARIOUS CARDIAC RHYTHMS ON PHYSIONET CHALLENGE DATABASE

Noise cleaning mechanism used	Median F_t, in 5 fold cross validation		
	SNR = 3 db	SNR = 0 db	SNR = -3 db
Bandpass filtering	0.62	0.48	0.43
Wavelet based noise removal	0.72	0.69	0.53
Arsene *et al.* [4] - CNN	0.82	0.74	0.63
Chiang *et al.* [5] - convolutional autoencoder	0.81	0.75	0.66
Proposed - sparse contractive autoencoder	**0.83**	**0.81**	**0.77**

obtained by the optimized model compared to the baseline model. Finally, a prototype end-to-end system is designed for real-time ECG de-noising using commercially available low-cost components. AD8232 [12] is a fully integrated portable single-lead ECG front-end. It has an operating voltage of 3.3 V and it comes with 3.5mm jack for connecting the electrodes to human body for measuring of ECG as analog voltage. The analog output is applied to a commercially available multi-channel 10-bit analog to digital converter (ADC), MCP3008. The ADC communicates with the General Purpose Input/Output (GPIO) pins of the Raspberry Pi 3, Model B+ via Serial Peripheral Interface (SPI) for synchronous data transfer. The Pi comes with 1.4 GHz 64-bit quad-core ARMv8 processor, 1 GB of memory and has 5 V operating voltage. The optimized model is copied to the Pi for inferencing. The users are asked to perform random body movement to introduce real motion artifacts in the recordings. The noisy ECG stream is digitalized and accumulated at the Pi at a sampling rate of 100 Hz and the de-noising is performed on every 1024 samples. Two buffers are used in the application to speed up the processing. Data in one buffer is being processed while the next stream of data is read into the other buffer in real-time. Fig. 5 shows a noisy ECG stream recorded by the sensor and the corresponding cleaned output. Depending upon the number of heart beats, the de-noising time for 1024 ECG samples (10.24 seconds) is measured as 41 ± 6 milliseconds on the target device.

TABLE II
MEAN RMSE IN RECONSTRUCTION OF ECG ON THE TARGET EDGE DEVICE AFTER OPTIMIZING THE BASELINE MODEL AT SNR = -3, 0, 3 DB

Test dataset name	Mean RMSE by the baseline autoencoder	Mean RMSE by the proposed optimized autoencoder
Simulated Dataset	0.08	0.11
PhysioNet Database	0.16	0.22

V. CONCLUSION

An effective noise cleaning of ECG is an important pre-processing step for designing an automatic disease diagnosis algorithm. A novel architecture of a convolutional de-noising autoencoder is proposed in this paper. The proposed approach is capable of efficiently reconstruction of clean ECG data at various SNR levels and the same is found to outperform a number of signal processing and deep learning based prior approaches. We also show that the reconstructed data can be successfully used for classifying abnormal heart rhythms. The proposed model is duly optimized for real-time noise cleaning of ECG on low-power edge devices using commercially available sensors. In future, we aim to further optimize the model to effectively run on smaller microcontroller units along with integrating various disease classification algorithms to realize continuous on-device ECG monitoring system.

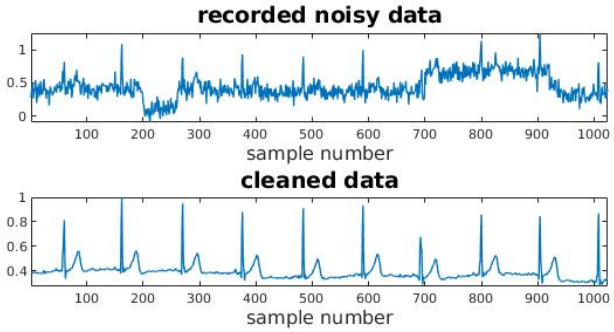


Fig. 5. Real-time noise cleaning of ECG on Raspberry Pi

REFERENCES

[1] G. Lu, J.-S. Brittain, P. Holland, J. Yianni, A. L. Green, J. F. Stein, T. Z. Aziz, and S. Wang, "Removing ecg noise from surface emg signals using adaptive filtering," *Neuroscience letters*, vol. 462, no. 1, pp. 14–19, 2009.

[2] J. Kuzilek, V. Kremen, F. Soucek, and L. Lhotska, "Independent component analysis and decision trees for ecg holter recording de-noising," *PLoS One*, vol. 9, no. 6, 2014.

[3] J. Lee, D. D. McManus, S. Merchant, and K. H. Chon, "Automatic motion and noise artifact detection in holter ecg data using empirical mode decomposition and statistical approaches," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1499–1506, 2011.

[4] C. T. Arsene, R. Hankins, and H. Yin, "Deep learning models for denoising ecg signals," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[5] H.-T. Chiang, Y.-Y. Hsieh, S.-W. Fu, K.-H. Hung, Y. Tsao, and S.-Y. Chien, "Noise reduction in ecg signals using fully convolutional denoising autoencoders," *IEEE Access*, vol. 7, pp. 60 806–60 813, 2019.

[6] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[7] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," 2011.

[8] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[9] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. Smith, "A dynamical model for generating synthetic electrocardiogram signals," *IEEE transactions on biomedical engineering*, vol. 50, no. 3, pp. 289–294, 2003.

[10] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark, "Af classification from a short single lead ecg recording: The physionet computing in cardiology challenge 2017," *Proceedings of Computing in Cardiology*, vol. 44, p. 1, 2017.

[11] S. Datta, C. Puri, A. Mukherjee, R. Banerjee, A. D. Choudhury, R. Singh, A. Ukil, S. Bandyopadhyay, A. Pal, and S. Khandelwal, "Identifying normal, af and other abnormal ecg rhythms using a cascaded binary classifier," *Computing*, vol. 44, p. 1, 2017.

[12] "Ad8232 ecg sensor datasheet," https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf, accessed: 2021-09-30.