IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 5, 2020, accepted July 19, 2020, date of publication July 29, 2020, date of current version August 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3012904

Detecting Noisy ECG QRS Complexes Using WaveletCNN Autoencoder and ConvLSTM

BROSNAN YUEN, XIAODAI DONG, (Senior Member, IEEE), AND TAO LU, (Member, IEEE)
Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada

Corresponding authors: Xiaodai Dong (xdong@ece.uvic.ca) and Tao Lu (taolu@ece.uvic.ca)

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN-2018-03778 and Grant RGPIN-2020-05938, in part by the Defense Threat Reduction Agency under Grant HDTRA11810047, and in part by the Undergraduate Student Research Awards.

ABSTRACT In this paper, we propose a novel machine learning pipeline to detect QRS complexes in very noisy wearable electrocardiogram (ECG) devices. The machine learning pipeline consists of a Butterworth filter, two wavelet convolutional neural networks (WaveletCNNs) autoencoders, an optional QRS complex inverter, a Monte Carlo k-nearest neighbours (k-NN), and a convolutional long short-term memory (ConvLSTM). WaveletCNN autoencoders filter out electrode contact noise, instrumentation noise, and motion artifact noise by using the advantages of wavelet filters and convolutional neural networks. The QRS complex inverter flips inverted QRS complexes. Monte Carlo k-NN performs automatic gain control on the ECG signals in order to normalize it. The ConvLSTM executes the final QRS complex detection by using the power of a convolutional neural network and a long short-term memory. The MIT-BIH, the European ST-T, and the Long Term ST database Noise Stress Test databases provide the training and testing ECG recordings. The proposed machine learning pipeline performs 3 standard deviations better than the state of the art QRS complex detection algorithms in terms of F_1 score for very noisy environments.

INDEX TERMS Artificial neural networks, electrocardiogram (ECG), QRS complex, feedforward neural networks, multi-layer neural network, convolutional neural networks, recurrent neural networks.

I. INTRODUCTION

Many cardiovascular diseases are diagnosed using electrocardiogram (ECG) recordings. For example, cardiovascular diseases such as coronary artery disease, arrhythmia, and heart valve disease are detected using ECG recordings. However, accurate diagnoses of cardiovascular diseases require large amounts of ECG recordings. Wearable ECG devices were invented in order to gather numerous amounts of ECG recordings for the cardiologists. The downside to the vast stores of ECG recordings is the disease classification time. As the number of ECG recordings increases, the amount of time the cardiologists spend on disease classification increases.

Automated ECG disease classification was created to expedite the cardiologists' diagnoses. More recently, deep neural networks were applied to ECG disease classification. Zihlmann *et al.* [1] proposed a convolutional neural network (CNN) followed by a long short-term memory (LSTM) network for ECG disease classification. Andersen *et al.* [2] developed a CNN-LSTM network that can detect atrial fibrillation (AF) in real time as it can process 24 h recordings in less than 1 second. Furthermore, a CNN-LSTM is created by Verma and Agarwal [3] for classifying between normal, AF, noisy signals, and other signals. Pourbabaee *et al.* [4] tested many CNNs with support vector machines (SVMs) and multilayer perceptron (MLPs) for paroxysmal atrial fibrillation detection. The CNN-LSTM generally performs better than the MLP and the SVM because the former is able to detect spatial-temporal patterns in the ECG signal.

In order to diagnose cardiovascular diseases using the ECG recordings, the ECG recordings must first be segmented. QRS complex detection is required for ECG segmentation because the QRS complexes mark the starts and the ends of the cardiac rhythm measurements and ECG segments. As a result, QRS complex detection is paramount for beat classification.

Many QRS complex detection algorithms have been created over the past few decades. The literature [5], [6] and references therein show a vast array of QRS complex detection algorithms. Most of the QRS complex detection algorithms use digital filters to detect the QRS complexes. The

The associate editor coordinating the review of this manuscript and approving it for publication was Thomas Canhao Xu.

main digital filters [5], [6] for QRS complex detection are the amplitude filter, the time series filter, the matched filter, and the frequency filter. Amplitude filters use thresholds to find the QRS complexes. If an ECG peak is above a threshold, then the amplitude filter classifies the peak as a QRS complex. Certain portions of Pan and Tompkins [7], GQRS [8], and ecgpuwave [9] employ the amplitude filter. The algorithms presented above perform within 0.1% of each other under low noise conditions. As a result, the algorithms have reached the Bayes error rate for QRS complex detection.

The matched filter convolves a predefined template with the ECG signal in order to produce a QRS complex detection signal. These papers [10], [11] show classical applications of the match filters for QRS complex detection. On the other hand, more advanced neural network based match filters have been developed. These references [12]–[20] contain neural network based match filters. The frequency filters allow for decomposition of signals based on frequency and time. As a result, certain frequencies could be attenuated in order to reduce the noises. Moreover, the frequency filters allow for analysis of QRS complexes as QRS complexes have unique frequency signatures. This improves the QRS complex detection accuracy. Wavedet [21] and WQRS [22] are examples of ECG frequency filters. Other examples of frequency filters include [23]–[32].

The QRS complex detection algorithms presented above are well suited for detecting QRS complexes in clean ECG signals. However, the algorithms above perform poorly in very noisy wearable ECG devices. Noises [33] such as baseline wandering, power line noise, electrode contact noise, and instrumentation noise create many problems. Baseline wandering introduces many low frequency noises to the ECG signal, which distorts the amplitudes of the ECG peaks. The other noises introduce high frequency noises, which creates false QRS complexes. Moreover, the high frequency noises make the actual QRS complexes undetectable by disfiguring them.

This paper proposes a novel machine learning pipeline to solve the problems above. The machine learning pipeline consists of a Butterworth filter, two wavelet convolutional neural network (WaveletCNN) autoencoders, an optional QRS complex inverter, a Monte Carlo k-nearest neighbours (k-NN), and a convolutional long short-term memory (ConvLSTM). The Butterworth filter removes the baseline wandering of the ECG signals by attenuating the low frequency noise components. The WaveletCNN Autoencoders are advanced bandpass filters, which essentially eliminate the false QRS complexes and enhance the actual QRS complexes. A QRS complex inverter is used to flip the inverted QRS complexes for better detection. The Monte Carlo k-NN applies automatic gain control to the ECG signals in order to normalize the peaks of the ECG signals to 1 mV. The ConvLSTM executes the final QRS complex detection by using the advantages of a CNN and a LSTM. As a result of the machine learning pipeline, the detection of QRS complexes in noisy wearable ECG devices is feasible.

The rest of the paper is organized as follows. Section II discusses several related QRS complex detection algorithms in detail. Section III shows the data preparation and the test environment. The proposed machine learning pipeline is presented in Section IV. Section V compares the performance metrics of the proposed machine learning pipeline to the other QRS complex detection algorithms. Finally, conclusions are given in Section VI.

II. RELATED QRS COMPLEX DETECTION ALGORITHMS

In this section, the following related QRS complex detection algorithms are presented: Pan and Tompkins [7], GQRS [8], Wavedet [21], Xiang *et al.*'s CNN [17], and Chandra *et al.*'s CNN [20]. The advantages and disadvantages of each algorithm are also described.

A. PAN AND TOMPKINS

The Pan and Tompkins algorithm [7] is the first real time QRS complex detection algorithm, in which a bandpass filter is applied to reduce the noises in the ECG signals, and adaptive filters are used to detect the QRS complexes. The adaptive filters consist of an amplitude filter, a slope filter, and a width filter. In order to be marked as a QRS complex, an ECG peak must simultaneously meet all of the following criteria: the peak's amplitude must be greater than an amplitude threshold, the peak's slope must be greater than a slope threshold, and the peak's width must fall within the range of a QRS complex width. The amplitude filter rejects the low amplitude signals, while the slope filter and the width filter eliminate the P waves and T waves. The advantages of the Pan and Tompkins algorithm are the fast processing times and low complexity. However, the filters used in the algorithm need to be engineered by hand, which requires a lot of time and expertise. Furthermore, the handcrafted filters can not adapt to different patients and environments.

B. GQRS

GQRS [8] is a classical QRS complex detection algorithm. Firstly, it calculates the means and the standard deviations of the RR intervals and the QRS complex amplitudes of the previously detected QRS. Secondly, the algorithm forms an adaptive search interval using the statistics of the RR intervals. Thirdly, the model creates an adaptive amplitude filter using the statistics of the QRS complex amplitudes. Finally, the adaptive amplitude filter is applied to the current adaptive search interval in order to detect the QRS complex. GQRS has the advantage of adapting slightly better than the Pan and Tompkins algorithm, which resulted in a better performance. However, GQRS still fails at detecting some of the QRS complexes because of its inability to adapt properly in noisy signals.

C. WAVEDET

Wavedet [21] is a wavelet based QRS complex detection algorithm. It performs wavelet decomposition on the ECG

signals, which produces a time series of frequencies. After the decomposition, a matched filter detects the QRS complexes by looking at the patterns of the wavelet coefficients. The matched filter allows for the analysis of many different signals at varying frequencies and time intervals, thus enabling the separation of the QRS complex signals from the non QRS complex signals. For the final QRS complex detection, it uses an adaptive amplitude filter. Wavedet performs better than GQRS under low noise conditions due to its multi-resolution analysis but performs poorly under high noise conditions due to its ineffective matched filter and adaptive amplitude filter. The matched filter is unable to filter out the noises as it can not distinguish the false QRS complexes from the actual QRS complexes. Furthermore, the amplitude filter can not tell the difference between the noises and the actual QRS complexes just by looking at the amplitudes.

D. AUTOMATIC QRS COMPLEX DETECTION USING TWO-LEVEL CONVOLUTIONAL NEURAL NETWORK

Xiang *et al.*'s paper [17] detects QRS complexes using a 2-layer CNN. The first ECG channel is obtained by applying a difference filter to the original input ECG signal. The second ECG channel is produced by applying a moving average filter and a difference filter to the original input ECG signal. After filtering, two 1×5 pixel CNN kernels are applied to the ECG channels. For the second CNN layer, it uses a 1×5 pixel CNN kernel. Finally, the MLP layers make the final QRS complex predictions. Xiang *et al.*'s CNN is fast and produces great results under low noise conditions. However, Xiang *et al.*'s CNN is ineffective under high noise conditions due to its difference filter. The difference filter is a highpass filter that allows high frequency noise through, which introduces classification errors and decreases the performance of the algorithm.

E. ROBUST HEARTBEAT DETECTION FROM MULTIMODAL DATA VIA CNN-BASED GENERALIZABLE INFORMATION FUSION

Chandra *et al.*'s paper [20] uniquely features an inter-patient testing scheme. In the testing scheme, the patients in the training set differ from the patients in the testing set. This testing scheme proves the generalization ability of their algorithm. Their neural network has a 1-layer CNN and an MLP. The CNN has 2 filters with a kernel size of 29 pixels. The MLP has one 200-neuron hidden layer and employs a sigmoid activation function. The model performs slightly better than Xiang *et al.*'s CNN due to the former's large CNN kernel size and the former's greater number of neurons. However, it was not designed for high noise conditions, and hence its performance degrades in very noisy data that often happen in wearable ECG devices.

III. DATA PREPARATION

As stated in the introduction, data preparation provides the testing and training environment to compare the various QRS complex detection algorithms. The MIT-BIH arrhythmia database [34], [35] (`https://physionet.org/content/mitdb/1.0.0/`), the European ST-T database [36] (`https://physionet.org/content/edb/1.0.0/`), and the Long Term ST database [37] (`https://physionet.org/content/ltstdb/1.0.0/`) are selected for the training and testing of the QRS complex detection algorithms. Noise is added to the ECG recordings using the PhysioToolkit Noise Stress Test [38] (`https://physionet.org/content/nstdb/1.0.0/`).

A. PRE-PROCESSING PROCEDURE

The following labels are selected for QRS complex detection: N, •, L, R, A, a, J, S, V, F, e, j, E, /, f, and Q. Some of the ECG recordings in the databases have inconsistent label positioning. A portion of the QRS complexes are labeled at the R peak, while other QRS complexes are labeled at the start of the Q wave. For this paper, the QRS complexes labeled at the R peak are used. For every individual sample that has a QRS complex label, $y = 1.0$ is assigned to that individual sample, which usually corresponds to the R peak position or very close to the R peak. The floats $y = 0.0$ are assigned to all other samples in the recording. There is only one $y = 1.0$ label for each QRS complex.

All detection algorithms are restricted to using only the primary ECG lead for QRS complex detection, while other ECG leads are not used. The usage of only the primary ECG lead is done to mimic wearable single channel ECG devices. The datasets are trained and tested on a patient level. Patients with multiple ECG recordings in the database had only one ECG recording included in this study. The total dataset contains N number of patients. For training, the total dataset is randomly shuffled and X number of patients are randomly selected for the training dataset. The remaining $N - X$ number of patients are used for the testing dataset. This way the patients from the training dataset differ from the patients in the testing dataset, minimizing bias towards the training dataset. The process above repeats for 10 times for 1×10 fold testing.

B. PhysioToolkit NOISE STRESS TEST

The datasets have relatively clean ECG recordings. To simulate the noisy wearable ECG devices, noise is added to the ECG recordings using the PhysioToolkit Noise Stress Test [38] (NST). The NST produces baseline wandering noises, white Gaussian noises, muscle noise, and electrode contact noises based on the given signal to noise ratio (SNR). Baseline wandering noises consist of low frequency high amplitude noises, while electrode contact noises are high frequency signals that look similar to QRS complexes. In this paper, only the first 640,000 samples of each ECG recording are used due to the constraints of the NST. The worst case SNR for most wearable ECG devices ranges from 12 dB SNR to -6 dB SNR. As a result, only the 12 dB SNR, the 0 dB SNR, and the -6 dB SNR ECG recordings are used.

C. MIT-BIH NST

For the MIT-BIH NST, the following recordings are used for training and testing: 100, 104, 108, 113, 117, 122, 201, 207, 212, 217, 222, 231, 101, 105, 109, 114, 118, 123, 208, 213, 219, 223, 232, 102, 106, 111, 115, 119, 124, 203, 209, 214, 220, 228, 233, 103, 107, 112, 116, 121, 200, 205, 210, 215, 221, 230, and 234. The QRS complex inverter is applied to the MIT-BIH as it has many inverted QRS complexes. For the training and the testing process, 17 random patient recordings are used as the training dataset and the remaining 30 patient recordings are grouped as the testing dataset.

D. EUROPEAN ST-T NST

The MIT-BIH database is sampled at 360 Hz, or equivalently 1 sample per 2.78 ms. In order to maintain a consistent sample rate, the European ST-T database is upsampled from 250 Hz to 360 Hz. Furthermore, the following ECG recordings from the European ST-T database are used for training and testing: e0103, e0104, e0111, e0112, e0113, e0115, e0116, e0118, e0123, e0127, e0136, e0147, e0151, e0154, e0159, e0161, e0166, e0170, e0203, e0204, e0206, e0207, e0208, e0210, e0212, e0303, e0306, e0404, e0406, e0408, e0409, e0410, e0411, e0417, e0418, e0509, e0601, e0606, e0607, e0609, e0610, e0611, e0612, e0613, e0615, e0704, e0818, and e1304. The QRS complex inverter is not used on the European ST-T database as most of European ST-T's QRS complexes are not inverted. For the training and the testing process, 14 random patient recordings are used as the training dataset and the remaining 34 patient recordings are grouped as the testing dataset.

E. LONG TERM ST NST

Similar to the European ST-T database, the Long Term ST database is upsampled from 250 Hz to 360 Hz. Furthermore, the following ECG recordings from the Long Term ST database are used for training and testing: s20011, s20031, s20091, s20101, s20111, s20131, s20151, s20201, s20211, s20231, s20251, s20261, s20271, s20281, s20291, s20321, s20331, s20341, s20351, s20361, s20371, s20401, s20411, s20431, s20451, s20461, s20471, s20521, s20551, s20631, s30671, s30701, s30741, and s30801. The QRS complex inverter is not used on the Long Term ST database. For the training and the testing process, 14 random patient recordings are used as the training dataset and the remaining 20 patient recordings are grouped as the testing dataset.

IV. PROPOSED MACHINE LEARNING PIPELINE

Noisy ECG signals introduce large amounts of errors into the QRS complex detection process. Noises such as baseline wandering, electrode contact noise, and instrumentation noise are present in ECG signals. This paper proposes a novel machine learning pipeline to denoise the ECG signals and to detect the QRS complexes.

Fig. 1 shows the machine learning pipeline for detecting QRS complexes. The input ECG signal contains low frequency baseline wandering signals. Firstly, the Butterworth high pass filter eliminates the baseline wandering signals by attenuating the low frequency signals. This effectively stabilizes the ECG signal. Secondly, the WaveletCNN Autoencoder 1 filters out the non QRS complex signals. WaveletCNN Autoencoder 1 applies wavelet decomposition to the ECG signal in order to obtain the wavelet coefficients. After that, the wavelet coefficients feeds into a CNN autoencoder. The CNN autoencoder filters the noisy wavelet coefficients and produces the clean wavelet coefficients. Subsequently, wavelet reconstruction recovers the clean single channel ECG signal from the clean wavelet coefficients. Thirdly, the difference filter sharpens and enhances the QRS complexes. This is done by attenuating the low frequency components, while maintaining the high frequency components resembling the QRS complexes. Fourthly, the ECG signal passes through the WaveletCNN Autoencoder 2. The WaveletCNN Autoencoder 2 performs the same filtering operations as the WaveletCNN Autoencoder 1. For some datasets, a QRS complex inverter is applied to flip the inverted QRS complexes. Fifthly, the Monte Carlo k-NN normalizes the ECG signals by scaling the ECG peaks to 1 mV.

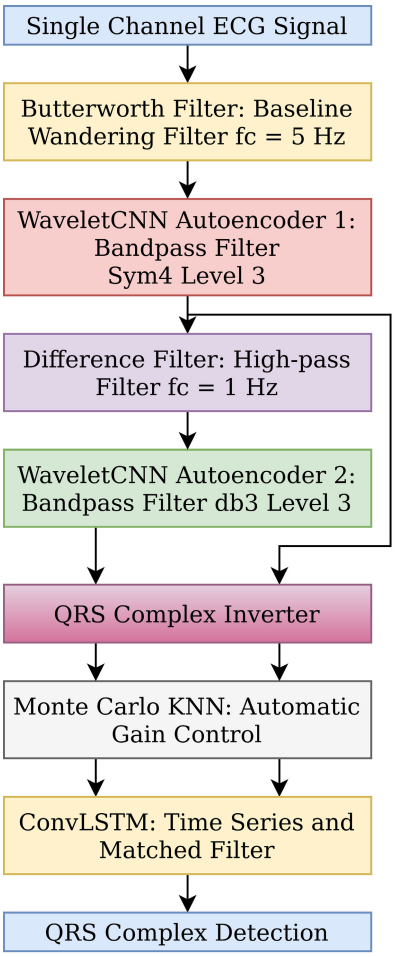

FIGURE 1. Machine learning pipeline for detecting QRS complexes.

Normalization is required in order to compensate for the differing patients and ECG devices. Lastly, the ConvLSTM uses the filtered ECG signals to predict the QRS complexes. The ConvLSTM uses the timing accuracy of a LSTM combined with the pattern matching ability of a CNN to detect spatial temporal signals such as the QRS complexes.

A. BUTTERWORTH FILTER: BASELINE WANDERING FILTER

Baseline wandering signals create noises and distortions in the ECG signals. Furthermore, baseline wandering signals randomizes the amplitudes of the P waves, the R waves, and the T waves. As a consequence, the P waves and the T waves are often misclassified as the QRS complexes. In order to reduce the baseline wandering signals, a Butterworth highpass filter [39] of order $n = 3$ with $f_c = 5$ Hz is designed. The filter attenuates the low frequency baseline wandering signals, while preserving the high frequency components. The Butterworth filter essentially takes in a single channel baseline wandering ECG signal and outputs a single channel baseline removed ECG signal to the WaveletCNN Autoencoder 1.

B. WaveletCNN AUTOENCODER 1: BANDPASS FILTER

The QRS complex has a unique signature in the frequency and time domain. The wavelet filter [40] is able to isolate the QRS complexes from the other ECG signals by filtering out certain frequencies at certain time intervals. This lowers the false positive rate of the QRS complex detection algorithm. As a result, the algorithm achieves a higher detection accuracy.

Wavelet filtering starts with the wavelet decomposition process. Firstly, the approximation coefficients $A[t]$ are calculated by convolving the input function $x[t]$ with the lowpass filter $g[t]$, i.e.,

$$A[t] = \sum_{k=-\infty}^{\infty} x[k]g[t-k] = x[t] * g[t]. \tag{1}$$

Secondly, the detail coefficients $D[t]$ are computed by convolving the input function $x[t]$ with the highpass filter $h[t]$, i.e.,

$$D[t] = \sum_{k=-\infty}^{\infty} x[k]h[t-k] = x[t] * h[t]. \tag{2}$$

Thirdly, both sets of coefficients $A[t]$, $D[t]$ are downsampled by 2. That is,

$$A_{down}[t] = A[2t+1] \tag{3}$$

$$D_{down}[t] = D[2t+1] \tag{4}$$

Finally, the downsampled detail coefficients $D_{down}[t]$ are kept, while the downsampled approximation coefficients $A_{down}[t]$ are fed as input $x[t]$ into the next wavelet level. The process above repeats until the wavelet coefficients of the desired level are obtained.

For every wavelet level, the detail coefficients $D_{down}[t]$ represent a fixed frequency component of the ECG signal. By attenuating certain detail coefficients $D_{down}[t]$, the non QRS complex signals such as instrumentation noises are reduced. This improves the performance of the proposed machine learning pipeline. Furthermore, certain wavelet coefficients are amplified in order to sharpen the QRS complexes. The process effectively decreases false negative rate and increases the true positive rate.

One way to filter the wavelet coefficients is by using the CNN [41]. The wavelet coefficients could be viewed as a 2D image, of which the CNN filters. The CNN removes the noises by detecting invalid spatial patterns in the wavelet coefficients and eliminating them. The CNN equations shown in

$$V^i_{y,z} = \sum_{j=1}^{N}\sum_{k=1}^{M} W^i_{j,k} X^i_{j+y-1,k+z-1} \tag{5}$$

$$O^i = A(V^i + B^i) \tag{6}$$

depict 2D convolution between the input matrix X^i and the kernel weights W^i. The kernel weights W^i slide across the input data X^i to produce V^i. After computing V^i, V^i is added to bias B^i. Then the resulting matrix passes through the activation function A and produces the output matrix O^i.

The WaveletCNN Autoencoder 1 receives the single channel baseline removed ECG signal from the Butterworth filter and outputs a filtered single channel ECG signal. The WaveletCNN Autoencoder 1 effectively functions as a bandpass filter by removing the false P waves, R waves, and T waves.

Fig. 2 shows the architecture of the WaveletCNN Autoencoder 1. Firstly, the WaveletCNN Autoencoder 1 performs wavelet decomposition on the noisy single channel ECG signal in order to produce the wavelet coefficients. WaveletCNN Autoencoder 1 uses the symlets 4 level 3 wavelet for wavelet decomposition. The symlets 4 wavelet approximates the shape of the QRS complex, which allows for the analyses of the QRS complexes. Secondly, the CNN encoder filters the wavelet coefficients using the 3×51 CNN kernels. After filtering, the CNN encoder downsamples the wavelet coefficients. The downsampling forces the WaveletCNN Autoencoder 1 to extract the most important features and to eliminate the unnecessary wavelet coefficients.

Thirdly, the CNN decoder reconstructs the original wavelet coefficients from the downsampled wavelet coefficients. The decoder uses the 3×51 kernel transpose CNN to upsample the wavelet coefficients. After upsampling, the wavelet coefficients are filtered by a CNN again. Fourthly, the neural network soft thresholding layer applies smoothing to the wavelet coefficients. Smoothing removes the high frequency noise. Finally, WaveletCNN Autoencoder 1 performs wavelet reconstruction on the filtered wavelet coefficients. In the end, a clean single channel ECG signal is produced.

Table 1 shows the hyper-parameter tuning of the WaveletCNN Autoencoder 1. The kernel size of the CNN layers is varied until the optimal kernel size of 3×51 is determined. Afterwards, the optimal number of channels is

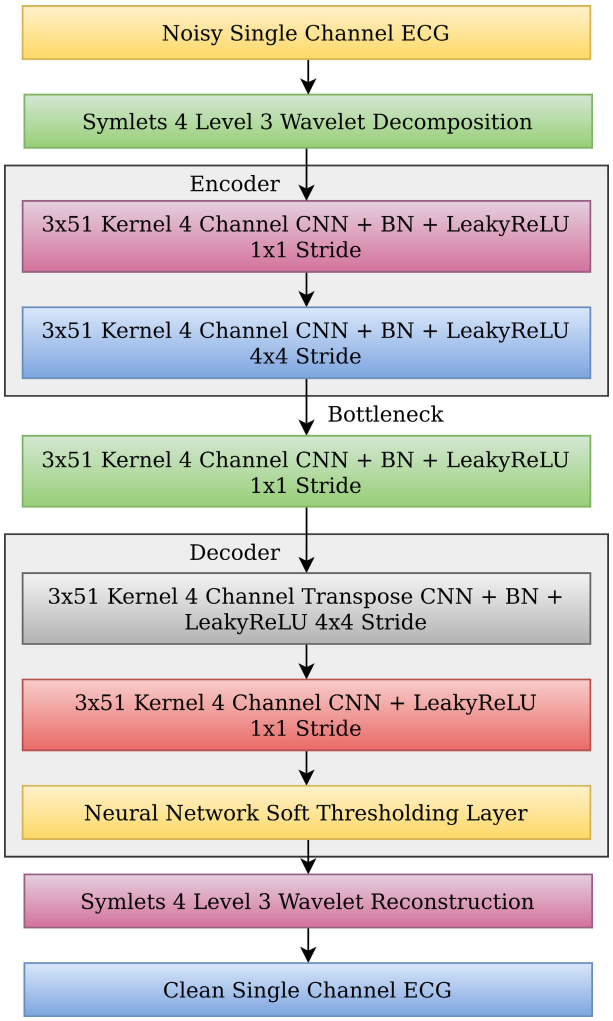

FIGURE 2. WaveletCNN Autoencoder 1 architecture.

found to be 4. The bottleneck stride controls the amount of filtering that happens on the wavelet coefficients. The optimal bottleneck stride is 4×4. Each CNN layer uses the LeakyReLU activation function given by

$$LeakyReLU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{7}$$

where x is the input matrix and α is the slope of the negative region. The LeakyReLU function prevents the loss function from reaching zero, which in turn prevents the optimizer getting stuck. The optimal slope determined by hyper-parameter tuning is $\alpha = 0.35$. Each CNN layer also uses the batch normalization function given by

$$BN(x) = \frac{x - \mu_x}{\sigma_x} \tag{8}$$

where μ_x, σ_x are the mean and the standard deviation of x respectively. Batch normalization helps the neural network to converge faster, which results in a faster training process. The neural network soft thresholding layer $\epsilon = 1 \times 10^{-6}$ shown

TABLE 1. Hyper-parameter tuning of WaveletCNN Autoencoder 1.

RMSE	Kernel Size	Channels	Bottleneck Stride	α
0.141181	3x21	4	4x4	0.35
0.138602	3x31	4	4x4	0.35
0.134275	3x41	4	4x4	0.35
0.120745	3x51	4	4x4	0.35
0.121464	3x61	4	4x4	0.35
0.129826	1x51	4	4x4	0.35
0.125492	2x51	4	4x4	0.35
0.120745	3x51	4	4x4	0.35
0.122911	4x51	4	4x4	0.35
0.140361	3x51	1	4x4	0.35
0.123508	3x51	2	4x4	0.35
0.124838	3x51	3	4x4	0.35
0.120745	3x51	4	4x4	0.35
0.141749	3x51	4	1x1	0.35
0.131968	3x51	4	2x2	0.35
0.120745	3x51	4	4x4	0.35
0.125231	3x51	4	4x4	0.01
0.122039	3x51	4	4x4	0.1
0.121347	3x51	4	4x4	0.2
0.120745	3x51	4	4x4	0.35
0.123383	3x51	4	4x4	0.5

Note: RMSE units in mV. MIT-BIH NST 0 dB SNR.

in

$$ReLU(x) = max(0, x) \tag{9}$$

$$\hat{y}_i = \frac{x}{|x| + \epsilon} ReLU(|x| - V_{thres}) \tag{10}$$

applies smoothing to the ECG signals. All input $|x|$ values are shifted down by the threshold V_{thres}. Subsequently, all input $|x|$ values below the threshold V_{thres} are set to zero. The WaveletCNN Autoencoder's output ECG signal $\hat{y}_i$ is produced by this layer.

The Adam optimizer [42] trains the WaveletCNN Autoencoder 1 using the root mean square error (RMSE) loss function stated below.

$$J(\hat{y}, y) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \tag{11}$$

where $\hat{y}_i$, y_i are the predicted and noiseless ECG samples respectively, and N is the number of ECG samples. The loss function forces the network to reconstruct a clean ECG signal from a noisy ECG signal. If the predicted de-noised ECG signal $\hat{y}$ differs from the ground truth ECG signal y, then the loss is large, otherwise the loss is small. Fig. 3 shows the WaveletCNN Autoencoder 1 removing noises from the single channel ECG signal.

C. DIFFERENCE FILTER: HIGH-PASS FILTER

In the ECG recordings, the R waves have sharper peaks than the P waves and the T waves. Therefore, the R waves have higher frequency components than the P waves and the T waves. The difference filter eliminates the P waves and the T waves by attenuating the low frequency components. As a result, the residual signal only consists of the high frequency

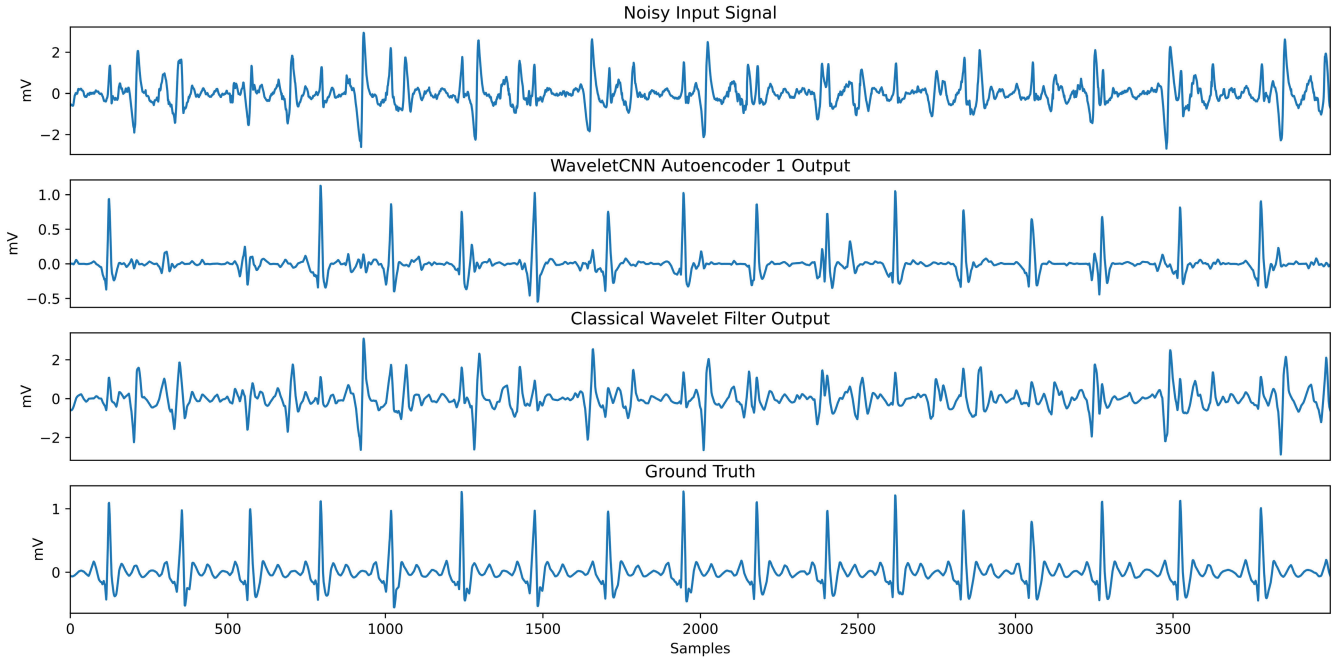

FIGURE 3. Comparison of the WaveletCNN Autoencoder 1 and the classical wavelet filter. MIT-BIH 0 dB test SNR.

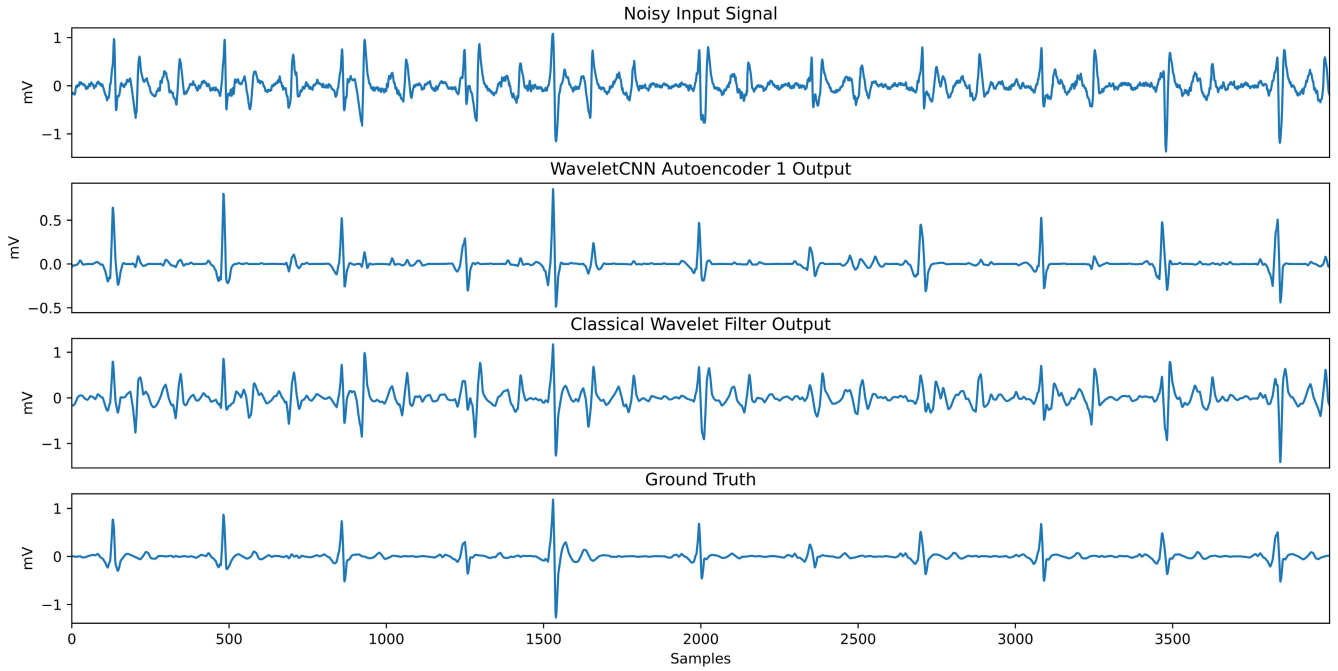

FIGURE 4. Comparison of the WaveletCNN Autoencoder 1 and the classical wavelet filter. MIT-BIH 0 dB test SNR.

components such as the R waves. The equation for the filter is presented below

$$y[t] = x[t] - x[t-1] \tag{12}$$

where $x[t]$ and $y[t]$ are the input and the output functions respectively. Alternatively, the filter could be thought as taking the gradient of the input function $x[t]$. In the end, the difference filter takes in the single channel ECG signal from the WaveletCNN Autoencoder 1 and sends the gradient of the ECG signal to the WaveletCNN Autoencoder 2.

D. WaveletCNN AUTOENCODER 2: BANDPASS FILTER

The WaveletCNN Autoencoder 2 takes in the single channel ECG signal from the difference filter and applies another

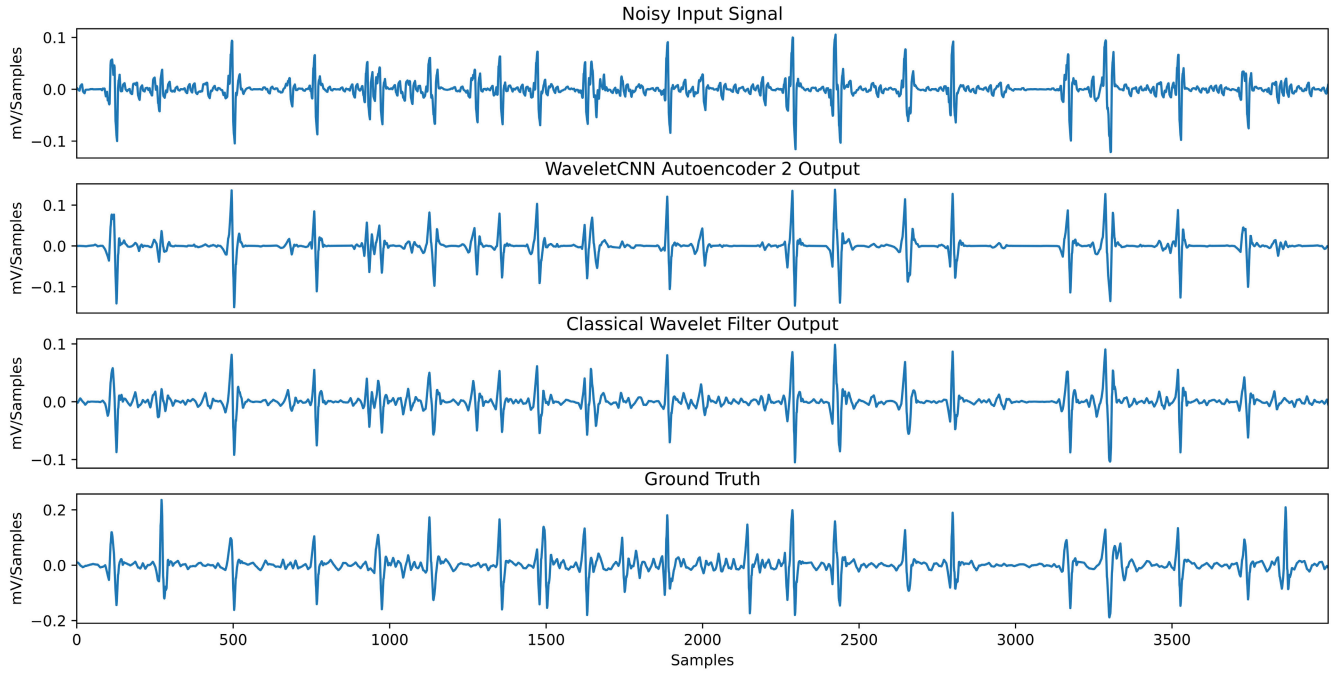

FIGURE 5. Comparison of the WaveletCNN Autoencoder 2 and the classical wavelet filter. MIT-BIH 0 dB test SNR.

layer of filtering. During filtering, it removes the noises from the ECG signals while preserving the gradient of the QRS complex. WaveletCNN Autoencoder 2 and WaveletCNN Autoencoder 1 have the exact same architecture. However, the wavelet scaling functions are different. WaveletCNN Autoencoder 2 uses the Daubechies 3 level 3 wavelet, which excels at capturing the gradient of the QRS complex. Fig. 5 shows the WaveletCNN Autoencoder 2 filtering the single channel ECG signal. The ground truth is obtained by taking the derivative of the original noiseless ECG signal.

E. QRS COMPLEX INVERTER (Optional)

Normal QRS complexes have positive R peak amplitudes. On the other hand, inverted QRS complexes have negative R peak amplitudes. Inverted QRS complexes create problems for the Monte Carlo k-NN gain control and the ConvLSTM detector because both of them expect the R peaks to be positive. An optional QRS complex inverter is created to flip the inverted QRS complexes. If a patient has many inverted QRS complexes, this method improves the detection accuracy, otherwise it introduces errors and lowers the detection accuracy.

The QRS complex inverter is very simple. It has a 60 sample wide window. If the absolute value of the minimum value of the window is greater than the maximum value of the window

$$|\min(window)| > \max(window) \tag{13}$$

the window moves to position of the minimum value, otherwise the window increments by 20 samples. If the window reaches a negative R peak, the window is inverted.

$$window \leftarrow -window \tag{14}$$

The process above repeats until the window reaches the end of the ECG recording. Fig. 6 shows the QRS complex inverter flipping inverted QRS complexes, while ignoring the normal QRS complexes.

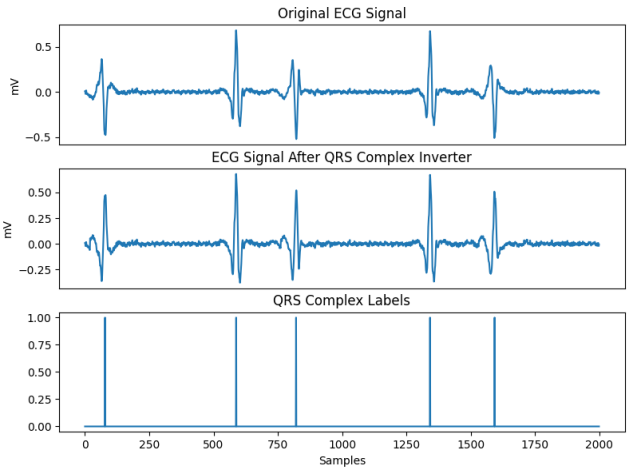

FIGURE 6. QRS complex inverter flipping inverted QRS complexes.

F. MONTE CARLO K-NN: AUTOMATIC GAIN CONTROL

Differing patients, instruments, and recording environments cause fluctuations in the QRS complex amplitudes. Normally, QRS complex detection algorithms use thresholds to detect

the QRS complexes. If a fluctuation causes a QRS complex amplitude to drop below the threshold, then the algorithm does not detect it. This creates a false negative error.

In order to minimize the fluctuations, the Monte Carlo k-NN normalizes the QRS complex amplitudes to 1 mV. Firstly, the window start variable *winstr* is randomly selected form a uniform distribution

$$winstr \in [0, len - winsize] \tag{15}$$

where *len* is the length of the input ECG signal $ECG1[t]$. Monte Carlo k-NN then calculates the window end

$$winend = winstr + winsize \tag{16}$$

using the window start and window size *winsize*. The window represents a time interval on the single channel ECG signal $ECG1[t]$. Secondly, k-NN clusters all the points in the window into two categories: R peak like signals and non R peak like signals. R peak like signals includes P waves, R waves, and T waves. The remaining signals represent the non R peak like signals. k-NN uses the Euclidean distances of the points' amplitudes for cluster assignment. Thirdly, the mean of the R peak like cluster

$$\mu_{Rpeak} = \frac{1}{N}\sum_{i=0}^{N} Rpeak_i \tag{17}$$

is computed, where $Rpeak_i$ is the amplitude of a R peak. Fourthly, the ECG signal between window start and window end

$$ECG1[winstr : winend] \leftarrow \frac{ECG1[winstr : winend]}{\mu_{Rpeak}} \tag{18}$$

is normalized using the R peak mean. Fifthly, the process repeats for X number of loops. The *winsize* slowly decreases by *decsize* for every Z loops. Finally, the single channel gain controlled $ECG2[t]$ signal is produced in a similar manner. Fig. 7 shows an example of the Monte Carlo k-NN with $X = 100len$ loops, $winsize = 2600$ initial window size, $decsize = 360$, and $Z = 20len$. In conclusion, the Monte Carlo k-NN takes in a single channel ECG signal from each autoencoder and outputs a 2 channel gain controlled ECG signal to the ConvLSTM.

G. ConvLSTM: TIME SERIES AND MATCHED FILTER

The filtering process above is good at cleaning and enhancing the ECG signals. However, classification errors still persist at this stage. Therefore, a final QRS complex detector is needed in order to reduce the classification errors. Using the ConvLSTM [43] layer, the combination of a CNN layer and a LSTM layer on the same layer, the spatial-temporal patterns of the QRS complexes are detected. The ConvLSTM layer performs better than the CNN-LSTM layers because the ConvLSTM detects spatial-temporal patterns concurrently instead of detecting spatial patterns in CNN layer and temporal patterns in the LSTM layer separately. Furthermore, the architectural design of the ConvLSTM is much simpler than the CNN-LSTM due the ConvLSTM needing less layers.

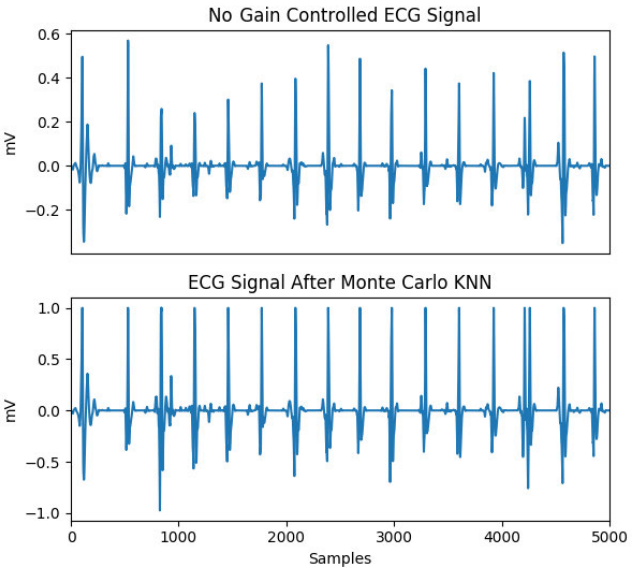

FIGURE 7. Application of the Monte Carlo k-NN.

Fig. 8 shows the ConvLSTM architecture, which has the same hyper-parameters as our previous paper [44]. The architecture consists of 1 CNN layer, 1 ConvLSTM layer, 1 max pooling layer, and 3 MLP layers. The CNN layer extracts features from the ECG signals, while filtering out the noises. After that, the ConvLSTM layer predicts the QRS complex timings. Subsequently, the max pooling layer reduces the 4 channel signal to a 1 channel signal. At the end, the MLP layers format the data and produce a pulse train marking the locations of the QRS complexes.

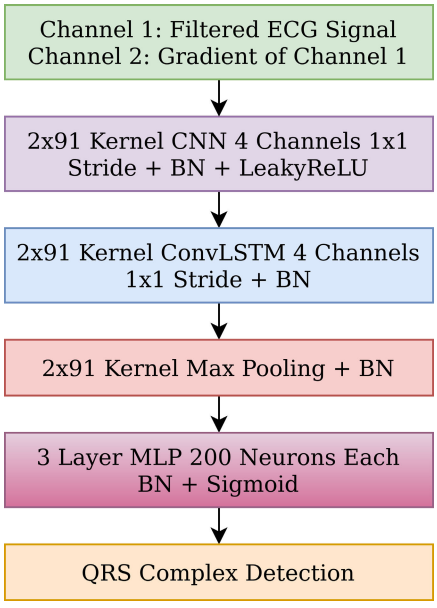

FIGURE 8. The ConvLSTM architecture.

The following paragraphs describe the ConvLSTM network in detail. Firstly, the network takes in the 2 channel

ECG signal from the Monte k-NN. Channel 1 is the normal filtered ECG signal, while channel 2 is the filtered gradient version of channel 1. Secondly, the CNN layer applies a 2×91 kernel with a 1×1 stride to the data. The 2×91 kernel matches the 91 sample long gradient of the QRS complex. The 1×1 stride preserves the timing information of the ECG signal. Moreover, the layer uses 4 channels because the ECG signals contain 4 main waveforms: P wave, QRS complex, T wave, and QS complex. The activation function is the LeakyReLU function with $\alpha = 0.02$. Thirdly, the ConvLSTM layer parses the features gathered by the CNN layer. The layer is a combination of the CNN layer and the LSTM layer. The ConvLSTM layer has the timing advantages of the LSTM, where it predicts the future QRS complex locations before they appear based on the previous locations. Moreover, the spatial detection abilities of the CNN allow the network to distinguish QRS complexes from the noises. The following equations show the functionality of the ConvLSTM.

$$S(x) = \frac{1}{1+e^{-x}} \tag{19}$$

$$i_t = S(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \tag{20}$$

$$f_t = S(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \tag{21}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{22}$$

$$o_t = S(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_{t-1} + b_o) \tag{23}$$

$$H_t = o_t \circ \tanh(C_t) \tag{24}$$

where W and b are the weight and bias matrices. $S(x)$ is the sigmoid activation function with respect to input matrix x. i_t, f_t, o_t, and H_t are the input gate, forget gate, hidden gate, and hidden state at time t respectively. X_t is the input image at time t and C_t is the output image at time t. $*$ denotes convolution and $\circ$ denotes element wise matrix multiplication. The ConvLSTM layer has 4 channels. Each channel has a 2×91 kernel with a 1×1 stride. The max pooling layer comes after the ConvLSTM layer, in which it selects the maximum value from the 2×91 image. Furthermore, the max pooling layer reduces the number of channels from 4 to 1 in order to speed up the network convergence. After that, the MLP layers execute the final QRS complex detection using the data from the max pooling layer. There are 3 MLP layers. Each MLP layer has 200 neurons. Each neuron uses the sigmoid activation function shown in Eqn. (19). The sigmoid activation function constrains the output of the network to the $\hat{y} \in [0, 1]$ interval. The ConvLSTM network produces a detection signal $\hat{y}_t$ with respect to time t. If the ConvLSTM detects a QRS complex, then it predicts $\hat{y}_t = 1.0$, otherwise it predicts $\hat{y}_t = 0.0$. The ConvLSTM uses the weighted cross-entropy loss function

$$J(\hat{y}, y) = -\log(S(\hat{y}))(y)(W_{pos}) - \log(1 - S(\hat{y}))(1-y) \tag{25}$$

for training, where y is the actual QRS complex and W_{pos} is the cross-entropy weight. The cross-entropy weight is $W_{pos} = 300$ because the RR interval is approximately 300 samples wide. Fig. 9 shows the ConvLSTM predicting a QRS complex using both ECG channels.

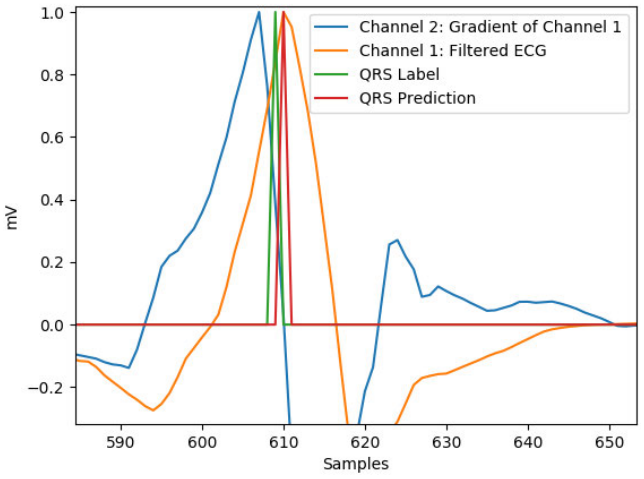

FIGURE 9. ConvLSTM QRS complex prediction using both ECG channels.

V. SIMULATIONS

In this paper, a few algorithms described in Section I are implemented for comparison to our machine learning pipeline. The QRS complex detection algorithms are benchmarked using the noisy dataset described in Section III. The true positives (TP), false positives (FP), false negatives (FN), sensitivity (SEN), positive predictive value (PPV), F1 score (F_1), and RMSE timings of the QRS complex detection algorithms are recorded. Here, SEN, PPV and F_1 are computed according to the equations below,

$$SEN = \frac{TP}{TP + FN} \tag{26}$$

$$PPV = \frac{TP}{TP + FP} \tag{27}$$

$$F_1 = 2\frac{SEN \cdot PPV}{SEN + PPV}. \tag{28}$$

Sensitivity measures the number of false positives in the predicted QRS complexes. Subsequently, positive predictive value measures the number of false negatives. If a QRS complex detection algorithm performs well, then it must have a high sensitivity $SENS \approx 1$ and a high positive predictive value $PPV \approx 1$. This in turn causes the $F_1 \approx 1$ to be high. If a QRS complex algorithms performs poorly then all the metrics are low $SENS \approx 0$, $PPV \approx 0$, $F_1 \approx 0$.

If a QRS complex detection algorithm predicts a QRS complex within 50 ms of a true QRS complex, then the prediction counts as a true positive. If a QRS complex detection algorithm predicts a QRS complex and a true QRS complex does not exist within 50 ms of the predicted QRS complex, then the predicted QRS complex counts as a false positive. If a QRS complex detection algorithm does not predict a QRS complex within 50 ms of a true QRS complex, then the true QRS complex counts as a false negative. The true negatives are not relevant as none of the ECG metrics use them.

Table 2 shows the RMSE of the WaveletCNN Autoencoder 1 and the classical wavelet filter. Table 2 proves

TABLE 2. WaveletCNN Autoencoder 1 and classical wavelet RMSE.

SNR	WaveletCNN Autoencoder 1	Classical Wavelet Filter
12 dB	0.0617 ± 0.010	0.0702 ± 0.004
6 dB	0.0917 ± 0.016	0.1476 ± 0.020
0 dB	0.1158 ± 0.020	0.2829 ± 0.026
-6 dB	0.1623 ± 0.024	0.5737 ± 0.026

Note: MIT-BIH NST. Confidence interval of 2σ. Units in mV.

TABLE 3. WaveletCNN Autoencoder 2 and Classical Wavelet RMSE.

SNR	WaveletCNN Autoencoder 2	Classical Wavelet Filter
12 dB	0.0129 ± 0.002	0.0112 ± 0.001
6 dB	0.0169 ± 0.001	0.0173 ± 0.001
0 dB	0.0206 ± 0.002	0.0214 ± 0.002
-6 dB	0.0301 ± 0.010	0.0317 ± 0.004

Note: MIT-BIH NST. Confidence interval of 2σ. Units in $\frac{mV}{Samples}$.

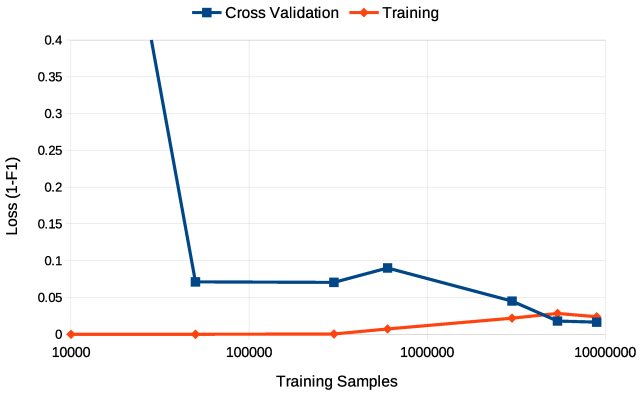

FIGURE 10. ConvLSTM's learning curve. MIT-BIH NST 12 dB SNR.

the WaveletCNN Autoencoder 1 filters out noises better than the classical wavelet filter because the WaveletCNN Autoencoder 1 has a lower RMSE mean. The performance improvement of the WaveletCNN Autoencoder 1 is due to the complex CNN autoencoder structure. Table 3 shows the WaveletCNN Autoencoder 2 performs slightly worse than the classical wavelet filter as the WaveletCNN Autoencoder 2 has a higher RMSE mean. The result is caused by the ECG signal after the difference filter being relatively clean and the WaveletCNN Autoencoder 2 introducing slight noises.

Fig. 10 shows the learning curve of the ConvLSTM. The ConvLSTM approximately converges at 4480000 training samples. The minimum number of training samples is at least 4480000 training samples, of which 4480000 training samples translates to 7 different ECG recordings.

The Bayes factor K [45], [46] is a statistical tool that compares the relative probabilities of the alternative hypothesis H_1 and the null hypothesis H_0. The Bayes factor K is defined by

$$K = \frac{P(D|H_1)}{P(D|H_0)} = \frac{P(H_1|D)}{P(H_0|D)} \frac{P(H_0)}{P(H_1)} \tag{29}$$

and is calculated using the observed data D in Tables 4-9, which show the classification performances of the QRS complex detection algorithms. Alternative hypothesis H_1 assumes the proposed method's F_1 score is higher than the other QRS complex algorithms' F_1 scores. Null hypothesis H_0 assumes the other QRS complex algorithms' F_1 scores are higher than the proposed method's F_1 score. Assume the prior probabilities of the hypotheses are equal $P(H_0) = P(H_1)$. Assume the F_1 scores follow a truncated Gaussian distribution in the range of [0, 1].

Table 10 shows the Bayes factors K for each QRS complex algorithm comparison. Two algorithms are compared at the same time and one algorithm is always the proposed method. Table 11 shows the degree of evidence against the null hypothesis H_0. For the MIT-BIH NST 12 dB SNR, all the Bayes factors are above $K > 3.2$. According to the Table 11, there is substantial evidence against the null hypothesis H_0. Therefore, the null hypothesis H_0 is rejected. There is a lack of strong or decisive evidence against the null hypothesis H_0 because the variances in Xiang *et al.*'s F_1 score [17] and Chandra *et al.*'s F_1 score [20] are large. Similar to MIT-BIH NST 12 dB SNR, all the Bayes factors K in MIT-BIH NST 0 dB SNR are above $K > 3.2$. As a result, the null hypothesis H_0 is rejected. For the comparison between the proposed method and Xiang *et al.*, there is only substantial evidence against the null hypothesis H_0 because Xiang *et al.*'s F_1 score has high variance.

In the European ST-T NST 12 dB SNR, all the Bayes factors are $K > 10$ except for Chandra *et al.*'s F_1 score. This is due to the database having a high SNR and Chandra *et al.*'s method having a similar performance to the proposed method. The null hypothesis H_0 is rejected due to all Bayes factors being $K > 3.2$ and having substantial evidence against the null hypothesis H_0. When the SNR decreases in the European ST-T NST 0 dB SNR, all the Bayes factors are $K > 1000$. As a result, there is decisive evidence against the null hypothesis H_0 and the null hypothesis H_0 is rejected without a shadow of a doubt. In the Long Term ST NST 0 dB SNR and the Long Term ST NST -6 dB SNR, all the Bayes factors are $K > 10$. There is strong evidence against the null hypothesis H_0 and the null hypothesis H_0 is rejected.

Reasons for the machine learning pipeline performing better than the other methods are presented below. Baseline wandering noises have frequencies below 5 Hz, of which can be reduced using a highpass filter that removes all signals below 5 Hz. On the other hand, wavelets are basically the templates of the targeted signal that the user wants to detect. To detect a specific signal, the wavelet is convolved with the ECG recording. The positions of the targeted signals can be determined by looking at the local maximums of the convolved signal. Most targeted signals do not have a single frequency. In order to detect targeted signals at different frequencies, the wavelet is first convolved with ECG recording. After convolution, the resulting signal is down-sampled to detect low frequency signals. Convolution in the first step is

TABLE 4. MIT-BIH NST 12 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
GQRS [8]	65189 ± 2738	12916 ± 1568	3526 ± 555	0.9486 ± 0.008	0.8345 ± 0.021	0.8879 ± 0.014	12.32 ± 0.16
Pan-Tompkins [7]	54866 ± 3735	1175 ± 328	14077 ± 4563	0.7961 ± 0.061	0.9789 ± 0.006	0.8778 ± 0.038	6.97 ± 0.57
Wavedet [21]	66837 ± 2553	14077 ± 1511	1829 ± 551	0.9733 ± 0.007	0.8259 ± 0.020	0.8935 ± 0.012	4.29 ± 0.42
Xiang et al. [17]	58484 ± 31567	4402 ± 3057	10349 ± 31103	0.8490 ± 0.454	0.9284 ± 0.032	0.8640 ± 0.371	3.75 ± 5.03
Chandra et al. [20]	60972 ± 27072	8745 ± 6331	6365 ± 25768	0.9036 ± 0.393	0.8708 ± 0.085	0.8743 ± 0.285	4.06 ± 4.76
Proposed	65751 ± 2935	3148 ± 1489	1980 ± 687	0.9707 ± 0.010	0.9543 ± 0.021	0.9624 ± 0.013	2.98 ± 0.71

Note: Confidence interval of 2σ. Timing RMSE units in samples.

TABLE 5. MIT-BIH NST 0 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
GQRS [8]	54774 ± 3361	35214 ± 1680	12977 ± 919	0.8083 ± 0.016	0.6085 ± 0.025	0.6943 ± 0.021	12.12 ± 0.10
Pan-Tompkins [7]	11343 ± 1517	9440 ± 580	57163 ± 1823	0.1654 ± 0.017	0.5452 ± 0.043	0.2538 ± 0.025	6.05 ± 0.28
Wavedet [21]	56384 ± 2011	29608 ± 849	12565 ± 869	0.8177 ± 0.010	0.6556 ± 0.013	0.7277 ± 0.011	4.89 ± 0.36
Xiang et al. [17]	51730 ± 35180	21180 ± 15049	16559 ± 34578	0.7566 ± 0.508	0.7387 ± 0.181	0.6923 ± 0.463	3.12 ± 1.63
Chandra et al. [20]	59868 ± 2820	24388 ± 6237	8809 ± 1349	0.8717 ± 0.019	0.7112 ± 0.060	0.7831 ± 0.041	3.63 ± 0.67
Proposed	62917 ± 2462	12806 ± 3724	5330 ± 1324	0.9219 ± 0.018	0.8311 ± 0.044	0.8739 ± 0.024	3.86 ± 0.63

Note: Confidence interval of 2σ. Timing RMSE units in samples.

TABLE 6. European ST-T NST 12 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
GQRS [8]	65835 ± 2726	9511 ± 897	1206 ± 599	0.9819 ± 0.008	0.8737 ± 0.011	0.9247 ± 0.009	12.66 ± 0.30
Pan-Tompkins [7]	55556 ± 4387	2241 ± 713	11517 ± 2522	0.8281 ± 0.037	0.9612 ± 0.011	0.8896 ± 0.024	10.11 ± 0.55
Wavedet [21]	66374 ± 2135	18451 ± 1752	1616 ± 352	0.9761 ± 0.005	0.7824 ± 0.020	0.8686 ± 0.014	11.81 ± 0.34
Xiang et al. [17]	58909 ± 8037	2283 ± 1666	6752 ± 8403	0.8974 ± 0.126	0.9633 ± 0.022	0.9277 ± 0.063	0.75 ± 0.37
Chandra et al. [20]	66101 ± 2813	3293 ± 1338	297 ± 145	0.9955 ± 0.002	0.9525 ± 0.019	0.9735 ± 0.010	0.83 ± 0.22
Proposed	65696 ± 1736	2137 ± 676	628 ± 413	0.9905 ± 0.006	0.9685 ± 0.009	0.9794 ± 0.006	1.46 ± 0.23

Note: Confidence interval of 2σ. Timing RMSE units in samples.

TABLE 7. European ST-T NST 0 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
GQRS [8]	58611 ± 1680	32359 ± 980	8354 ± 854	0.8752 ± 0.012	0.6442 ± 0.012	0.7421 ± 0.011	12.38 ± 0.16
Pan-Tompkins [7]	15671 ± 2187	8101 ± 762	53038 ± 2386	0.2280 ± 0.030	0.6586 ± 0.035	0.3386 ± 0.038	12.03 ± 0.39
Wavedet [21]	57813 ± 1419	31697 ± 711	9913 ± 624	0.8536 ± 0.007	0.6458 ± 0.010	0.7353 ± 0.007	12.35 ± 0.38
Xiang et al. [17]	57121 ± 6175	17375 ± 7378	8880 ± 4375	0.8650 ± 0.069	0.7683 ± 0.076	0.8130 ± 0.054	1.75 ± 0.45
Chandra et al. [20]	62234 ± 2475	19569 ± 3184	3882 ± 829	0.9412 ± 0.012	0.7609 ± 0.032	0.8414 ± 0.019	1.73 ± 0.23
Proposed	63130 ± 1382	9991 ± 3527	2922 ± 514	0.9557 ± 0.007	0.8637 ± 0.043	0.9072 ± 0.023	1.98 ± 0.46

Note: Confidence interval of 2σ. Timing RMSE units in samples.

TABLE 8. Long term ST NST 0 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
GQRS [8]	36393 ± 1457	22031 ± 611	6458 ± 265	0.8492 ± 0.005	0.6228 ± 0.013	0.7186 ± 0.010	12.08 ± 0.35
Pan-Tompkins [7]	8054 ± 1828	4975 ± 485	35151 ± 3286	0.1867 ± 0.047	0.6165 ± 0.050	0.2862 ± 0.060	6.16 ± 0.36
Wavedet [21]	37534 ± 1617	17973 ± 969	5958 ± 325	0.8629 ± 0.005	0.6761 ± 0.021	0.7581 ± 0.014	4.95 ± 0.22
Xiang et al. [17]	39488 ± 3270	10615 ± 2243	2937 ± 2394	0.9305 ± 0.058	0.7887 ± 0.024	0.8533 ± 0.017	1.65 ± 0.11
Chandra et al. [20]	40114 ± 1261	9605 ± 1409	2208 ± 586	0.9478 ± 0.013	0.8069 ± 0.023	0.8716 ± 0.014	1.60 ± 0.12
Proposed	41205 ± 1647	5075 ± 1946	1450 ± 346	0.9659 ± 0.008	0.8904 ± 0.040	0.9265 ± 0.024	1.74 ± 0.31

Note: Confidence interval of 2σ. Timing RMSE units in samples.

repeated again to detect lower and lower frequency signals. White Gaussian noise is present in all frequencies. However, electrode contact noise only appears at high frequencies and has a unique shape. Furthermore, the QRS complex only appears at high frequencies and also has a unique shape. The wavelet filter can effectively apply a bandpass filter that eliminates all other frequencies except for the frequencies that the QRS complex resides in. This will remove the white Gaussian noise outside the QRS complex frequency range. Moreover, the wavelet could be chosen to only match the QRS complex and not the other signals, thus removing the electrode contact noise.

TABLE 9. Long term ST NST −6 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
GQRS [8]	32520 ± 824	28147 ± 276	10285 ± 323	0.7597 ± 0.003	0.5360 ± 0.006	0.6285 ± 0.004	12.07 ± 0.30
Pan-Tompkins [7]	2412 ± 73	7685 ± 294	40508 ± 1397	0.0562 ± 0.002	0.2389 ± 0.007	0.0910 ± 0.002	9.84 ± 0.16
Wavedet [21]	33985 ± 1491	22126 ± 657	9657 ± 476	0.7787 ± 0.003	0.6056 ± 0.016	0.6813 ± 0.010	5.29 ± 0.21
Xiang et al. [17]	35727 ± 2218	18910 ± 2549	6397 ± 2184	0.8481 ± 0.051	0.6541 ± 0.033	0.7384 ± 0.033	2.54 ± 0.19
Chandra et al. [20]	34898 ± 1093	19016 ± 1214	7292 ± 655	0.8272 ± 0.012	0.6473 ± 0.018	0.7262 ± 0.011	2.71 ± 0.08
Proposed	37774 ± 1827	14343 ± 3373	5011 ± 1078	0.8828 ± 0.025	0.7252 ± 0.052	0.7962 ± 0.042	2.41 ± 0.25

Note: Confidence interval of 2σ. Timing RMSE units in samples.

TABLE 10. Bayes factor K applied to Tables 4-9s' F_1 Scores.

Database	GQRS [8]	Pan-Tompkins [7]	Wavedet [21]	Xiang et al. [17]	Chandra et al. [20]
Proposed on MIT-BIH NST 12 dB SNR	> 1000	> 1000	> 1000	10.59	9.200
Proposed on MIT-BIH NST 0 dB SNR	> 1000	> 1000	> 1000	6.266	> 1000
Proposed on European ST-T NST 12 dB SNR	> 1000	> 1000	> 1000	23.56	5.421
Proposed on European ST-T NST 0 dB SNR	> 1000	> 1000	> 1000	> 1000	> 1000
Proposed on Long Term ST NST 0 dB SNR	> 1000	> 1000	> 1000	> 1000	> 1000
Proposed on Long Term ST NST -6 dB SNR	> 1000	> 1000	> 1000	64.62	> 1000

TABLE 11. Interpretation of Bayes factor K [45].

Bayes Factor K	Interpretation
1 to 3.2	Barely worth mentioning
3.2 to 10	Substantial evidence against the null hypothesis H_0
10 to 100	Strong evidence against the null hypothesis H_0
>100	Decisive evidence against the null hypothesis H_0

The WaveletCNN Autoencoder automatically determines which frequencies to filter and which signals to remove by adjusting the CNN filters. This is in contrast to the classical wavelet filter, where the frequency filters have to be chosen manually and they may not be optimal. Noises also cause fluctuations in the QRS complex amplitudes. Monte Carlo k-NN normalizes the QRS complex amplitudes to 1 mV. It does this by repeatedly scaling the random ECG windows using the mean value of the R peaks. This is in contrast to the traditional normalization techniques, where the ECG signal is only processed once, and it is not detailed enough.

For the ConvLSTM detector, it uses convolutions from the present window and convolutions from past windows to detect QRS complexes. ConvLSTM effectively uses temporal information as well as spatial information. For example, the ConvLSTM detects a QRS complex using the convolutions filters at window $t = 4s$. ConvLSTM knows the QRS complex period is approximately $\Delta t \approx 1\ s$. It can predict the next QRS complex will occur at approximately $t \approx 5s$. The ConvLSTM can confirm the prediction by executing a convolution on the window at $t = 5s$. This is in contrast to previous papers using CNNs, where the temporal information is not used, and prediction accuracy is inferior.

This paper is consistent with the previous works as the relative performances of the QRS complex algorithms remain the same. Pan-Tompkins [7] algorithm's F_1 score is drastically lower than the other algorithms' F_1 score under lower SNR conditions, because it is the first significant algorithm proposed and improvements were made by other work after its appearance.

VI. CONCLUSION

The proposed machine learning pipeline de-noises the ECG signals and detects the QRS complexes. The machine learning pipeline consists of a Butterworth filter, two WaveletCNNs autoencoders, a Monte Carlo k-NN, and a ConvLSTM. The Butterworth filter and the WaveletCNN autoencoders filter out the various noises in ECG signals. Monte Carlo k-NN normalizes the QRS complexes. The ConvLSTM executes the final QRS complex detection and produces the QRS complex detection signal. The MIT-BIH NST, the European ST-T NST, and the Long Term ST NST databases provide the training and testing ECG recordings. Null hypothesis H_0 assumes the other QRS complex algorithms' F_1 scores are higher than the proposed method's F_1 score. It has been demonstrated in Table 10 that all the Bayes factor $K > 3.2$, which means there is substantial evidence against the null hypothesis H_0. Therefore, the null hypothesis H_0 is rejected. In conclusion, the proposed machine learning pipeline outperforms existing QRS complex detection methods.

VII. LIMITATIONS AND FUTURE WORK

The proposed method is trained using the PhysioNet NST. As a result, the proposed method is optimized for the specific noise distribution present in the NST. However, the machine learning pipeline might have a lower accuracy when faced with real ECG noises as they might have different noise distributions. In the future, an online training version of this machine learning pipeline could be developed in order to

adapt to new noises in real time. This would allow the method to detect new QRS complexes beyond the initial training dataset. The QRS complex inverter and the Monte Carlo k-NN run very slowly because the scikit-learn [47] library only uses one CPU. The components could be parallelized in order to run on multiple CPUs. This will decrease the processing time.

Aside from the QRS complex, many other ECG waveforms are also important. For example, the T wave and P wave are very important for determining the ST interval, QT interval, and PR interval. The intervals are useful for classifying ECG diseases such as atrial fibrillation, arrhythmia, and branch blocks. Other biological signals like electroencephalogram (EEG) [48], electromyography (EMG), and photoplethysmogram (PPG) prove useful for diagnosing disorders and diseases. Similar to the ECG signals, the other biological signals are susceptible to noise. In the future, the machine learning pipeline could be used to reduce the noise, perform gain control, and detect the other waveforms.

APPENDIX
REVIEW ON NEURAL NETWORKS

Neural networks are the building blocks for some of the QRS complex detection algorithms. Therefore, a brief review of neural networks is presented below.

A. MULTI-LAYER PERCEPTRON

MLPs [49] are a type of neural network. They excel at classifying many types of data. Moreover, MLPs are proficient at adapting to changing input data and are suitable for detecting QRS complexes in ECG signals. The fundamental equation for the MLPs is given by

$$O_i = A(W_i^T X_i + B_i). \tag{30}$$

For each layer i, X_i, W_i, B_i are the input, weight, and bias matrices respectively. $A(V)$ is the activation function with respect to input matrix V. Also, O_i is the output matrix of layer i. The input data enters at the input matrix X_i. The input X_i is multiplied by the weights W_i and the result is added to the bias B_i. The output O_i is obtained by passing the result through the activation function $A(V)$.

B. LONG SHORT-TERM MEMORY

LSTMs [50] are a special type of neural network that stores memories inside of the neurons. LSTMs remember and forget data using the hidden gates. Therefore, LSTMs are suitable for time-series pattern recognition, where the LSTMs predict the future using only the past input data. The LSTM's equations are given by

$$i_t^l = S(W_i^l[\chi_t; h_{t-1}^l] + b_i^l) \tag{31a}$$
$$f_t^l = S(W_f^l[\chi_t; h_{t-1}^l] + b_f^l) \tag{31b}$$
$$s_t^l = f_t^l s_{t-1}^l + i_t^l \tanh(W_s^l[\chi_t; h_{t-1}^l] + b_s^l) \tag{31c}$$
$$o_t^l = S(W_o^l[\chi_t; h_{t-1}^l] + b_o^l) \tag{31d}$$
$$h_t^l = o_t^l \tanh(s_t^l) \tag{31e}$$

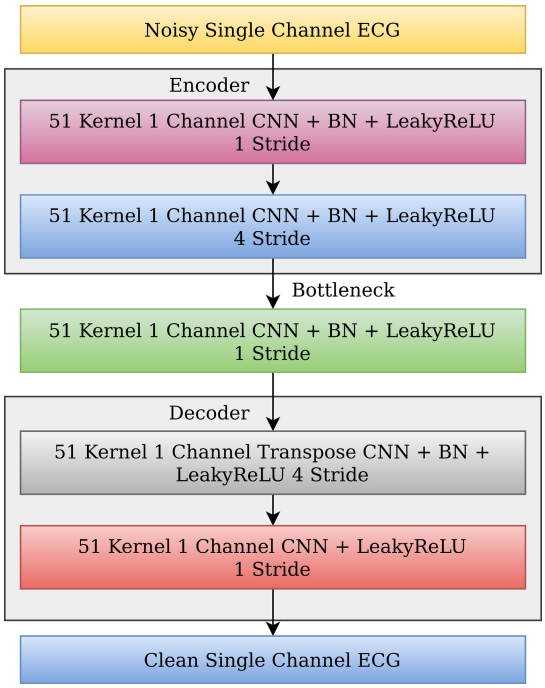

FIGURE 11. CNN Autoencoder 1 architecture.

TABLE 12. WaveletCNN Autoencoder 1 and CNN Autoencoder 1 RMSE.

SNR	WaveletCNN Autoencoder 1	CNN Autoencoder 1
12 dB	0.0617 ± 0.010	0.0824 ± 0.008
6 dB	0.0917 ± 0.016	0.1152 ± 0.020
0 dB	0.1158 ± 0.020	0.1442 ± 0.010
-6 dB	0.1623 ± 0.024	0.1754 ± 0.020

Note: MIT-BIH NST. Confidence interval of 2σ. Units in mV.

Eqn. (31a) shows the input gate for the LSTM neuron. Input gate controls which information enters the LSTM neuron. Input data χ_t and final LSTM output data h_{t-1}^l are fed into the input gate i_t^l. Using the input gate weights W_i^l and biases b_i^l, the output vector of the input gate i_t^l is determined. Eqn. (31b) shows the forget gate. The forget gate f_t^l determines if the hidden state s_t^l is forgotten or retained. The forget gate f_t^l is computed just like the input gate i_t^l. Eqn. (31c) shows the computation of hidden state vector s_t^l. The forget gate f_t^l controls the retention of the previous state variable s_{t-1}^l. The input gate i_t^l controls the weight of the tanh activation function. Eqn. (31d) shows the output gate o_t^l. The output gate o_t^l controls what information is outputted by the LSTM neuron. Eqn. (31e) shows the final output vector h_t^l of the LSTM neuron. Final output vector h_t^l is computed using the hidden state vector s_t^l and the output gate o_t^l.

C. COMPARISON OF WaveletCNN AUTOENCODER AND CNN AUTOENCODER

In this section, the WaveletCNN Autoencoder 1 is compared to the CNN Autoencoder 1. Fig. 11 shows the architecture of the CNN Autoencoder 1, of which is similar to the WaveletCNN Autoencoder 1. However, the wavelet decomposition and reconstruction are removed. Also, the CNN Autoencoder 1 only performs 1D convolutions on

TABLE 13. MIT-BIH NST 0 dB SNR algorithm performance.

Algorithm	TP	FP	FN	SENS	PPV	F_1 score	Timing RMSE
Xiang et al. [17] original 5 CNN kernel size	51730 ± 35180	21180 ± 15049	16559 ± 34578	0.7566 ± 0.508	0.7387 ± 0.181	0.6923 ± 0.463	3.12 ± 1.63
Xiang et al. [17] 30 CNN kernel size	58157 ± 5024	20307 ± 5655	10099 ± 4417	0.8519 ± 0.064	0.7418 ± 0.058	0.7925 ± 0.045	3.50 ± 0.36
Chandra et al. [20] original	59868 ± 2820	24388 ± 6237	8809 ± 1349	0.8717 ± 0.019	0.7112 ± 0.060	0.7831 ± 0.041	3.63 ± 0.67
Chandra et al. [20] with WaveletCNN Autoencoders	64856 ± 1984	25346 ± 7756	3320 ± 936	0.9512 ± 0.014	0.7202 ± 0.062	0.8194 ± 0.042	3.87 ± 0.36

Note: Confidence interval of 2σ. Timing RMSE units in samples.

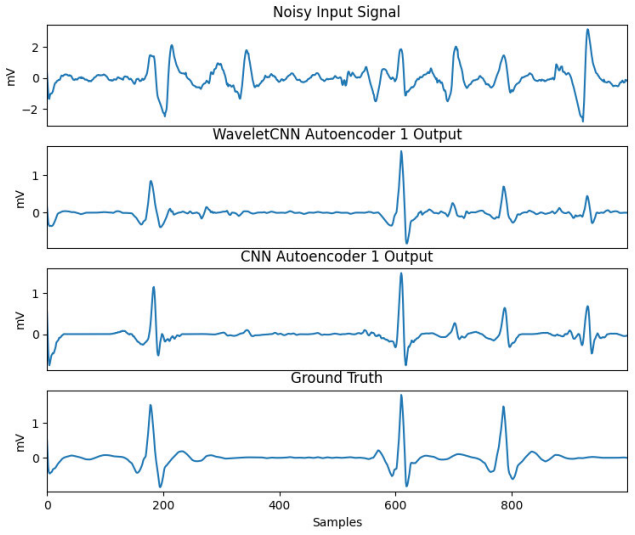

FIGURE 12. Comparison of WaveletCNN Autoencoder 1 and CNN Autoencoder 1. MIT-BIH NST 0 dB SNR.

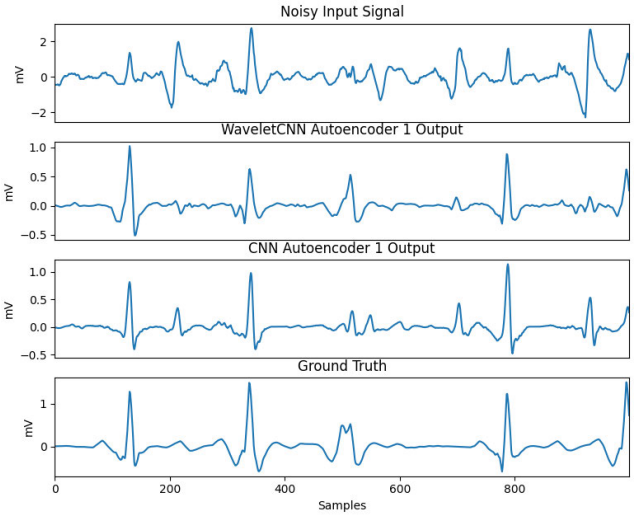

FIGURE 13. Comparison of WaveletCNN Autoencoder 1 and CNN Autoencoder 1. MIT-BIH NST 0 dB SNR.

the original 1D ECG signal instead of 2D convolutions on the 2D wavelet coefficients. Table 12 shows the comparison between the two encoders. For 12 dB SNR, the WaveletCNN Autoencoder 1's RMSE of 0.0617 mV is lower than the CNN Autoencoder 1's RMSE of 0.0824 mV. For -6 dB SNR, the WaveletCNN Autoencoder 1's RMSE of 0.1623 mV is within the variance of CNN Autoencoder 1's RMSE of 0.1754 mV. Fig. 12 and Fig. 13 show the output ECG signals of the WaveletCNN Autoencoder 1 and the CNN Autoencoder 1, where the WaveletCNN Autoencoder 1 is slightly better at filtering noises than the CNN Autoencoder 1. This is due to wavelet decomposition used in the WaveletCNN Autoencoder 1. The wavelet decomposition is scale invariant and extracts features at many different scales, of which allows the WaveletCNN to detect QRS complexes of various sizes. Moreover, the wavelet basis function allows the wavelet to distinguish the QRS complex from other ECG signals. As a result, the noise filtering capability is improved.

D. EFFECT OF CNN FILTERS ON PERFORMANCE

Table 13 shows the effect of the CNN filters on the QRS complex detection algorithms. When the CNN kernel size of the Xiang *et al.* [17] increases from 5 samples to 30 samples, the performance increases from $F_1 = 0.6923$ to $F_1 = 0.7925$. The larger CNN kernel sizes provide better performances in noisy environments because they are able to provide more information to the fully connected layers. This enables the fully connected layers to distinguish the QRS complex from the noises. When the WaveletCNN Autoencoders are used on Chandra *et al.* [20], the performance increases from $F_1 = 0.7831$ to $F_1 = 0.8194$. This is due to the WaveletCNN Autoencoder being able to filter out noises better than the normal CNN layers.

E. VENTRICULAR TACHYCARDIA

The machine learning pipeline is unable to detect ventricular tachycardia because the ventricular tachycardia has a very low frequency and the Butterworth highpass filter removes it. Moreover, the beats for ventricular tachycardia are labeled as "!" and they are not used in this study.

ACKNOWLEDGMENT

The GPU is sponsored by NVidia

REFERENCES

[1] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," in *Proc. Comput. Cardiol. Conf.*, Sep. 2017, pp. 1–4.
[2] R. S. Andersen, A. Peimankar, and S. Puthusserypady, "A deep learning approach for real-time detection of atrial fibrillation," *Expert Syst. Appl.*, vol. 115, pp. 465–473, Jan. 2019.
[3] D. Verma and S. Agarwal, "Cardiac arrhythmia detection from single-lead ECG using CNN and LSTM assisted by oversampling," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2018, pp. 14–17.

[4] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ECG features for screening paroxysmal atrial fibrillation patients," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2095–2104, Dec. 2018.

[5] B.-U. Kohler, C. Hennig, and R. Orglmeister, "The principles of software QRS detection," *IEEE Eng. Med. Biol. Mag.*, vol. 21, no. 1, pp. 42–57, Jan./Feb. 2002.

[6] G. M. Friesen, T. C. Jannett, M. A. Jadallah, S. L. Yates, S. R. Quint, and H. T. Nagle, "A comparison of the noise sensitivity of nine QRS detection algorithms," *IEEE Trans. Biomed. Eng.*, vol. 37, no. 1, pp. 85–98, Jan. 1990.

[7] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *IEEE Trans. Biomed. Eng.*, vol. BME-32, no. 3, pp. 230–236, Mar. 1985.

[8] Physionet. (2018). *GQRS, QRS Detector and Post-Processor*. [Online]. Available: https://www.physionet.org/physiotools/wag/gqrs-1.htm

[9] Physionet. (2018). *QRS Detection and Waveform Boundary Recognition Using Ecgpuwave*. [Online]. Available: https://www.physionet.org/physiotools/ecgpuwave/

[10] P. S. Hamilton and W. J. Tompkins, "Adaptive matched filtering for QRS detection," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Nov. 1988, pp. 147–148.

[11] D. T. Kaplan, "Simultaneous QRS detection and feature extraction using simple matched filter basis functions," in *Proc. Comput. Cardiol.*, 1990, pp. 503–506.

[12] G. Goovaerts, S. Padhy, B. Vandenberk, C. Varon, R. Willems, and S. Van Huffel, "A machine-learning approach for detection and quantification of QRS fragmentation," *IEEE J. Biomed. Health Informat.*, vol. 23, no. 5, pp. 1980–1989, Sep. 2019.

[13] S. S. Mehta and N. S. Lingayat, "SVM-based algorithm for recognition of QRS complexes in electrocardiogram," *IRBM*, vol. 29, no. 5, pp. 310–317, Nov. 2008.

[14] S. S. Mehta and N. S. Lingayat, "Identification of QRS complexes in 12-lead electrocardiogram," *Expert Syst. Appl.*, vol. 36, no. 1, pp. 820–828, Jan. 2009.

[15] Q. Xue, Y. H. Hu, and W. J. Tompkins, "Neural-network-based adaptive matched filtering for QRS detection," *IEEE Trans. Biomed. Eng.*, vol. 39, no. 4, pp. 317–329, Apr. 1992.

[16] B. Abibullaev and H. D. Seo, "A new QRS detection method using wavelets and artificial neural networks," *J. Med. Syst.*, vol. 35, no. 4, pp. 683–691, Aug. 2011.

[17] Y. Xiang, Z. Lin, and J. Meng, "Automatic QRS complex detection using two-level convolutional neural network," *Biomed. Eng. OnLine*, vol. 17, no. 1, p. 13, Jan. 2018, doi: 10.1186/s12938-018-0441-4.

[18] J. Camps, A. McCarthy, B. Rodrıguez, and A. Mincholé, "Deep learning based QRS multilead delineator in electrocardiogram signals," in *Proc. Comput. Cardiol.*, 2018, pp. 1–4.

[19] K. Arbateni and A. Bennia, "Sigmoidal radial basis function ANN for QRS complex detection," *Neurocomputing*, vol. 145, pp. 438–450, Dec. 2014.

[20] B. S. Chandra, C. S. Sastry, and S. Jana, "Robust heartbeat detection from multimodal data via CNN-based generalizable information fusion," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 3, pp. 710–717, Mar. 2019.

[21] J. P. Martinez, R. Almeida, S. Olmos, A. P. Rocha, and P. Laguna, "A wavelet-based ECG delineator: Evaluation on standard databases," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 570–581, Apr. 2004.

[22] W. Zong, G. Moody, and D. Jiang, "A robust open-source algorithm to detect onset and duration of QRS complexes," in *Proc. Comput. Cardiol.*, 2003, pp. 737–740.

[23] S. Kadambe, R. Murray, and G. F. Boudreaux-Bartels, "Wavelet transform-based QRS complex detector," *IEEE Trans. Biomed. Eng.*, vol. 46, no. 7, pp. 838–848, Jul. 1999.

[24] K. Mourad and B. R. Fethi, "Efficient automatic detection of QRS complexes in ECG signal based on reverse biorthogonal wavelet decomposition and nonlinear filtering," *Measurement*, vol. 94, pp. 663–670, Dec. 2016.

[25] C. Li, C. Zheng, and C. Tai, "Detection of ECG characteristic points using wavelet transforms," *IEEE Trans. Biomed. Eng.*, vol. 42, no. 1, pp. 21–28, Jan. 1995.

[26] F. Bouaziz, D. Boutana, and M. Benidir, "Multiresolution wavelet-based QRS complex detection algorithm suited to several abnormal morphologies," *IET Signal Process.*, vol. 8, no. 7, pp. 774–782, Sep. 2014.

[27] U. D. Ulusar, R. B. Govindan, J. D. Wilson, C. L. Lowery, H. Preissl, and H. Eswaran, "Adaptive rule based fetal QRS complex detection using Hilbert transform," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Sep. 2009, pp. 4666–4669.

[28] D. S. Benitez, P. A. Gaydecki, A. Zaidi, and A. P. Fitzpatrick, "A new QRS detection algorithm based on the Hilbert transform," in *Proc. Comput. Cardiol.*, 2000, pp. 379–382.

[29] F. I. de Oliveira and P. U. Cortez, "A QRS detection based on Hilbert transform and wavelet bases," in *Proc. 14th IEEE Signal Process. Soc. Workshop Mach. Learn. Signal Process.*, Sep. 2004, pp. 481–489.

[30] M. Jia, F. Li, J. Wu, Z. Chen, and Y. Pu, "Robust QRS detection using high-resolution wavelet packet decomposition and time-attention convolutional neural network," *IEEE Access*, vol. 8, pp. 16979–16988, 2020.

[31] C.-C. Lin, H.-Y. Chang, Y.-H. Huang, and C.-Y. Yeh, "A novel wavelet-based algorithm for detection of QRS complex," *Appl. Sci.*, vol. 9, no. 10, p. 2142, May 2019.

[32] M. B. Hossain, S. K. Bashar, A. J. Walkey, D. D. Mcmanus, and K. H. Chon, "An accurate QRS complex and p wave detection in ECG signals using complete ensemble empirical mode decomposition with adaptive noise approach," *IEEE Access*, vol. 7, pp. 128869–128880, 2019.

[33] H. Limaye and V. V. Deshmukh, "ECG noise sources and various noise removal techniques: A survey," *Int. J. Appl. Innov. Eng. Manage.*, vol. 5, no. 2, pp. 86–92, Feb. 2016.

[34] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, May 2001.

[35] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulat.*, vol. 101, no. 23, pp. E215–E220, 2000.

[36] A. Taddei, G. Distante, M. Emdin, P. Pisani, G. B. Moody, C. Zeelenberg, and C. Marchesi, "The European ST-T database: Standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography," *Eur. Heart J.*, vol. 13, no. 9, pp. 1164–1172, Sep. 1992.

[37] F. Jager, A. Taddei, G. B. Moody, M. Emdin, G. Antolič, R. Dorn, A. Smrdel, C. Marchesi, and R. G. Mark, "Long-term ST database: A reference for the development and evaluation of automated ischaemia detectors and for the study of the dynamics of myocardial ischaemia," *Med. Biol. Eng. Comput.*, vol. 41, no. 2, pp. 172–182, Mar. 2003.

[38] G. B. Moody, W. K. Muldrow, and R. G. Mark, "A noise stress test for arrhythmia detectors," *Comput. Cardiol.*, vol. 11, no. 3, pp. 381–384, 1984.

[39] S. Butterworth, "On the theory of filter amplifiers," *Exp. Wireless Wireless Eng.*, vol. 7, pp. 536–541, Oct. 1930.

[40] C. E. Heil and D. F. Walnut, "Continuous and discrete wavelet transforms," *SIAM Rev.*, vol. 31, no. 4, pp. 628–666, 1989.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[43] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[44] B. Yuen, X. Dong, and T. Lu, "Inter-patient CNN-LSTM for QRS complex detection in noisy ECG signals," *IEEE Access*, vol. 7, pp. 169359–169370, 2019.

[45] R. E. Kass and A. E. Raftery, "Bayes factors," *J. Amer. Statist. Assoc.*, vol. 90, no. 430, pp. 773–795, 1995.

[46] J. O. Berger and L. R. Pericchi, "The intrinsic Bayes factor for model selection and prediction," *J. Amer. Stat. Assoc.*, vol. 91, no. 433, pp. 109–122, Mar. 1996.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[48] R. K. Chikara and L.-W. Ko, "Neural activities classification of human inhibitory control using hierarchical model," *Sensors*, vol. 19, no. 17, p. 3791, 2019.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

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