Robust Peak Detection for Holter ECGs by Self-Organized Operational Neural Networks

Moncef Gabbouj, Serkan Kiranyaz, Junaid Malik, Muhammad Uzair Zahid, Turker Ince, Muhammad Chowdhury, Amith Khandakar, and Anas Tahir

Abstract— **Although numerous R-peak detectors have been proposed in the literature, their robustness and performance levels may significantly deteriorate in low-quality and noisy signals acquired from mobile electrocardiogram (ECG) sensors such as Holter monitors. Recently, this issue has been addressed by deep one-dimensional (1D) Convolutional Neural Networks (CNNs) that have achieved *state-of-the-art* performance levels in Holter monitors; however, they pose a high complexity level that requires special parallelized hardware setup for real-time processing. On the other hand, their performance deteriorates when a compact network configuration is used instead. This is an expected outcome as recent studies have demonstrated that the learning performance of CNNs is limited due to their strictly homogenous configuration with the sole linear neuron model. This has been addressed by Operational Neural Networks (ONNs) with their heterogenous network configuration encapsulating neurons with various non-linear operators. In this study, to further boost the peak detection performance along with an elegant computational efficiency, we propose 1D Self-Organized Operational Neural Networks (Self-ONNs) with generative neurons. The most crucial advantage of 1D Self-ONNs over the ONNs is their self-organization capability that voids the need to search for the best operator set per neuron since each generative neuron has the ability to create the optimal operator during training. The experimental results over the China Physiological Signal Challenge-2020 (CPSC) dataset with more than one million ECG beats show that the proposed 1D Self-ONNs can significantly surpass the *state-of-the-art* deep CNN with less computational complexity. Results demonstrate that the proposed solution achieves a 99.10% F1-score, 99.79% sensitivity, and 98.42% positive predictivity in the CPSC dataset which is the best R-peak detection performance ever achieved.**

Index Terms— **Convolutional Neural Networks, Operational Neural Networks, R-peak detection, Holter monitors.**

I. INTRODUCTION

ELECTROCARDIOGRAM (ECG) acquires the heartbeat sequence in time displaying the electrical depolarization-repolarization patterns of the heart. ECG signal forms itself in the QRS complexes and ventricular beats and it bears essential information about the status of the heart. Among many other tools, ECG is still the most significant non-invasive tool for cardiac monitoring and clinical diagnosis. R-peak detection is the primary operation that usually precedes any kind of ECG analysis such as ECG beat classification and cardiac arrhythmia detection [1]-[21]. Conventional Holter monitors and the recent introduction of low-cost and low-power mobile ECG sensors present a significant motive and challenge for robust and real-time detection of the R-peak locations. Especially, robustness is a key issue since it was reported in a recent study [22] that the R-peak detection performance can severely deteriorate when the ECG acquisition is poor and corrupted by a high level of noise.

Particularly for clinical ECG recordings, numerous R-peak detection algorithms have been proposed in the literature. One of the first and the most widely used algorithm was proposed by Pan and Tompkins [23], which has served as the benchmark method for more than three decades. Afterwards, several other popular methods based on signal processing have emerged such as Wavelet Transform [24], [25], Hilbert Transform [26], and ensemble empirical mode decomposition [27]. Some hybrid methods that consist of traditional signal processing and machine learning have followed, e.g., R-peak detection by Radial Basis Functions (RBFs) [28] and Hidden Markov Models (HMMs) [29]. The common approach in those classical detectors was to perform R-peak enhancement first by using signal processing techniques such as filter banks and spectral analysis and then applying the peak detection. A crucial advantage of such methods is that they are very fast. However, they are all designed for clinical ECG recordings with a clean, almost noise-free signal. All of them were evaluated on the benchmark MIT-BIH arrhythmia dataset [35] or other similar datasets with high-quality clinical ECG recordings. Their performance level significantly deteriorates when the ECG signal quality is poor [22] and thus they are not suitable for low-power mobile ECG sensors. In fact, a proper evaluation constitutes a major problem in general because even though there are few public datasets [30], [31] containing some noisy ECG signals with ground-truth R-peak locations, they are limited in size and duration. This problem has become the main bottleneck even for recent and modern peak detectors based on Deep Learning paradigms including

S. Kiranyaz, M. U. Zahid, M. E. H. Chowdhury, A. Khandakar, and A. Tahir, are with Electrical Engineering, College of Engineering, Qatar University, Qatar; e-mail: mkiranyaz@qu.edu.qa, m.uzair@qu.edu.qa, mchowdhury@qu.edu.qa, amitk@qu.edu.qa, a.tahir@qu.edu.qa.

T. Ince are with the Electrical & Electronics Engineering Department, Izmir University of Economics, Turkey; e-mail: turker.ince@izmirekonomi.edu.tr.

M. Gabbouj and O. C. Devecioglu is with the Department of Computing Sciences, Tampere University, Finland; e-mail: Moncef.gabbouj@tut.fi.

Long Short-Term Memory (LSTM) networks [32], and Convolutional Neural Networks (CNNs) [33], [34]. Both approaches in [32], [33] aimed to improve the robustness but only against the artificial (additive) noise. For this purpose, they have only induced certain additive noise such as baseline wander and motion artifacts to ECG records in the MIT-BIH dataset and reported a reasonable level of robustness. However, such artificial noise addition does not represent the actual degradations that occur in a Holter monitor where besides the severe and varying noise levels and occasional glitches, the baseline DC level varies drastically and frequently along with the dynamic range of QRS complexes as shown in Figure 1. Another issue in such evaluation over the MIT-BIH dataset is that this benchmark dataset does not contain a sufficient number of beats and variations where a deep network can truly be tested. This brings the danger of a certain bias and over-fitting over the results.
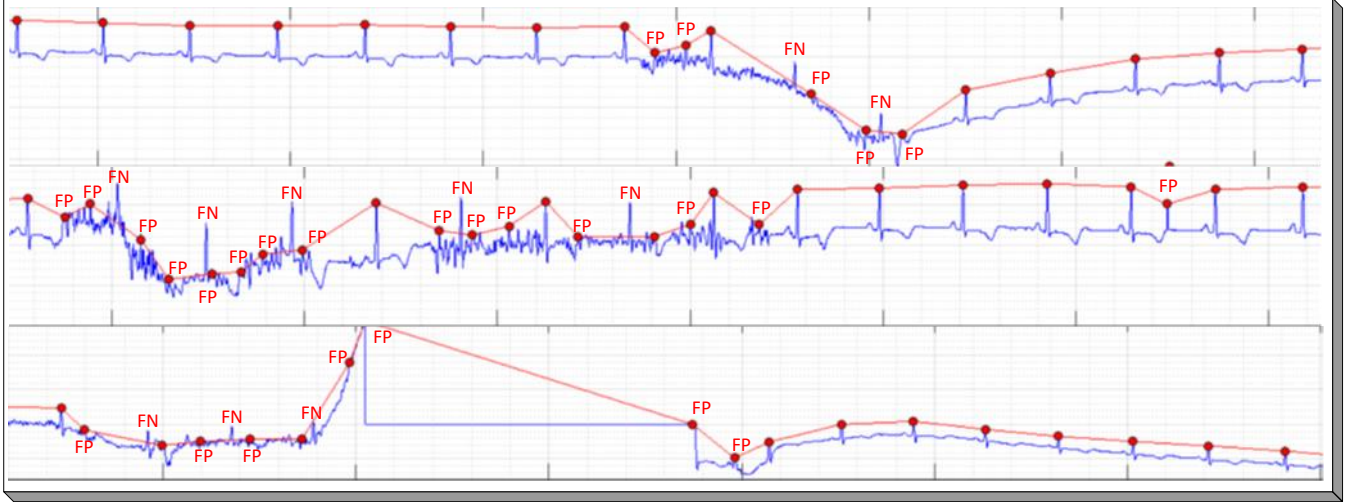

Figure 1: Typical Holter ECG segments from the record of patient 6 in the CPSC dataset. Red circles represent the R-peaks detected by the Pan-Tompkins method [23] where several false-positives (FPs) and false-negatives (FNs) are visible.

Recently, the study in [34] has proposed a novel method based on deep 1D CNNs that has actually been applied over a real Holter ECG repository, the China Physiological Signal Challenge (2020) dataset (CPSC) with more than 1 million ECG beats. This method has been tested over the entire dataset and achieved the *state-of-the-art* performance level for R-peak detection with a significant margin. However, it still suffers two major drawbacks: i) 12-layer deep model poses a significant computational complexity, and ii) both false-positives (FPs) and false-negatives (FNs) are still too high, especially on the arrhythmia beats. In this study, we address both issues with a novel network model.

According to the recent studies [42]-[50], the main drawback of the conventional Multi-Layer Perceptrons (MLPs) and their derivatives, CNNs, is that they both depend on the ancient neuron model (McCulloch-Pitts) [41]. It is neuroscientific fact that the mammalian neural systems are highly heterogeneous and consist of diverse (nonlinear) neuron types with specialized electrophysiological and biochemical properties [42], [43]. This linear neuron model is only a crude and simplified model of its biological counterpart. As a consequence, MLPs and CNNs with an entirely homogenous network configuration with linear neurons are capable of learning for relatively simple and linearly separable problems; however, they entirely fail to do so whenever the solution space of the problem is highly nonlinear and complex [42]-[50]. To address this major deficiency, Generalized Operational Perceptrons (GOPs) [42]-[49] and later on Operational Neural Networks (ONNs) [50] have been proposed. Both models are heterogeneous with any non-linear neuron model, and this gives them an elegant diversity level to learn highly complex and multi-modal functions or spaces with minimal network complexity and training data. The operational neurons in GOPs and ONNs mimic the biological neurons with, *nodal (*corresponding to the synaptic connections) and *pool* (corresponding to the integration in the soma) operators. An "operator set" is the combination of *nodal, pool,* and *activation* operators, and the operator set library is formed in advance to store all potential operator sets. However, ONNs can only achieve a limited heterogeneity level since one operator set has to be assigned to all neurons of each hidden layer. Furthermore, they are bound to the limited number of operators in the operator set library. The latter can be a serious bottleneck on the learning performance when the right operator set needed for the learning problem at hand is missing in the library. Finally, ONNs pose a high computational complexity especially during the training because the right operator sets have to be searched in advance with several Back-Propagation (BP) runs.

In this study, in order to address the aforementioned issues and drawbacks, we propose 1D Self-organized ONNs (Self-ONNs) with the *generative* neuron model for robust R-peak detection for Holter ECG in real-time. The generative neuron model enables the self-organization capability for Self-ONNs [51] - [55] where the nodal operators are iteratively generated during the BP training to maximize the learning performance. Obviously, the ability to create *any* non-linear nodal operator enables superior operational diversity and flexibility. So, Self-ONNs neither need an operator set library in advance, nor require any prior search process to find the optimal nodal

operator. 2D Self-ONNs have been proposed in recent studies [50] and [54] have shown that even with a few neurons they can achieve a superior learning performance for various image processing and regression tasks whilst the performance gap between ONNs and CNNs widens further. Hence, in this study, our primary goal is to achieve a superior R-peak detection performance compared to the deep 1D CNNs [34] while reducing the network complexity and depth significantly for a real-time application. Besides [34], we shall also perform comparative evaluations against the earlier state-of-the-art methods [23], [56]-[59] to accomplish an overall validation. As a summary, we can enlist the novel and significant contributions of this article as follows:

1- This is the first study where 1D Self-ONNs[1] have even been proposed for ECG peak detection evaluated over the largest ECG benchmark dataset the China Physiological Signal Challenge (2020) dataset (CPSC) with more than 1 million ECG beats.

2- Thanks to the heterogeneous network model with generative neurons, 1D Self-ONNs exhibit a superior learning capability that achieved the *state-of-the-art* peak detection performance even though it has half the network depth and more than four times fewer neurons.

3- The most important contribution of the proposed approach is the crucial reduction achieved on the missed (false-negatives) for arrhythmia beats compared to [34] and other major peak detection methods.

4- Along with superior R-peak detection performance compared to the deep 1D CNNs [34], the proposed 1D Self-ONNs enable a significant reduction in the depth and complexity of the deep CNN model proposed in [34].

5- Finally, this is the first article where the raw-vectorized BP formulations are presented for 1D Self-ONNs along with the corresponding computational complexity analysis.

The rest of the paper is organized as follows: Section II presents 1D Self-ONNs with generative neurons in detail and formulates the forward-propagation (FP) and back-propagation (BP) training. Section III first outlines the ECG datasets used in this study and presents the experimental setup used for testing and evaluation of the proposed R-peak detection approach based on 1D Self-ONNs. Then, the overall results are presented, and comparative evaluations using the standard performance metrics against several state-of-the-art techniques provided in Section IV. The computational complexity analysis of these networks is also presented in this section. Finally, Section V concludes the paper and suggests topics for future research.

II. 1D SELF-ORGANIZED OPERATIONAL NEURAL NETWORKS

In this section, we will proceed by revisiting how ONNs generalize the 1D convolution operation. Then the mathematical model of the proposed generative neuron-based 1D Self-ONN will be presented. To conclude, a simplification of the generative neuron will be discussed which can

significantly reduce the computational cost by enabling the use of fast vectorized operations.

NODAL TRANSFORMATIONS
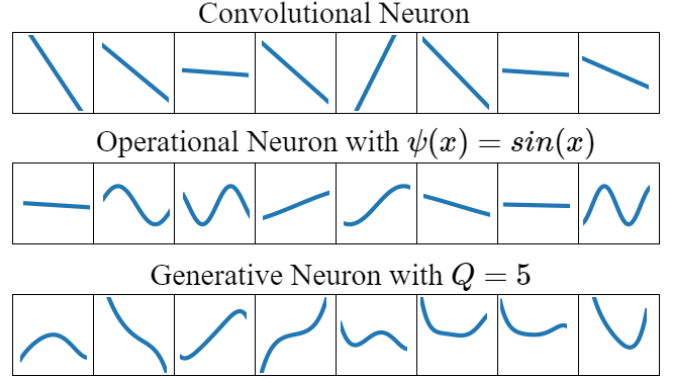
Figure 2: A visual comparison of different nodal transformation profiles entailed by the kernel of a convolutional, operational, and the proposed generative neuron of order Q. The generative neuron model enables enhanced nonlinearity and heterogeneity within the kernels.

ONNs are derived from the Generalized Operational Perceptrons (GOPs) in the same way CNNs are derived from MLPs with two restrictions: limited connectivity and weight sharing. GOPs have been proposed in [42], [45] to replace the basic (linear) neuron model from the 1950s (McCulloch-Pitts) [41] aiming to address the well-known limitations and drawbacks of MLPs. Recently, GOPs have outperformed not only MLPs but even the latest variants of Extreme Learning Machines (ELMs) [46]-[49]. Derived directly from GOPs, ONNs [50] are heterogeneous networks encapsulating neurons with linear and non-linear operators hence carrying a closer link to biological systems. In brief, ONNs extend the sole usage of linear convolutions in the convolutional neurons by the *nodal* and *pool* operators.

Let us consider the case of the kth neuron in the lth layer of a 1D CNN. For the sake of brevity, we assume the same convolution operation with unit stride and the required amount of zero-padding. The output of this neuron can be formulated as follows:

$$x_k^l = b_k^l + \sum_{i=0}^{N_{l-1}} x_{ik}^l \qquad (1)$$

where b_k^l is the bias associated with this neuron and x_{ik}^l is defined as:

$$x_{ik}^l = Conv1D(w_{ik}, y_i^{l-1}) \qquad (2)$$

Here, $w_{ik} \in \mathbb{R}^K$ is the kernel connecting the ith neuron of $(l-1)^{th}$ layer to the kth neuron of the lth layer, while $x_{ik}^l \in \mathbb{R}^M$ is the input map, and $y_i^{l-1} \in \mathbb{R}^M$ are the lth and $(l-1)^{th}$ layers' k^{th} and i^{th} neurons' outputs, respectively. By definition, the convolution operation of (2) can be expressed as,

---

$$x_{ik}^l(m) = \sum_{r=0}^{K-1} w_{ik}^l(r) y_i^{l-1}(m+r) \qquad (3)$$

The core idea behind an operational neuron is a generalization of the above as follows:

$$\overline{x_{ik}^l}(m) = P_k^l \left(\psi_k^l \left(w_{ik}^l(r), y_i^{l-1}(m+r) \right) \right)_{r=0}^{K-1} \qquad (4)$$

where $\psi_l^k(\cdot): \mathbb{R}^{M \times K} \to \mathbb{R}^K$ and $P_k^l(\cdot): \mathbb{R}^K \to \mathbb{R}^1$ are termed as *nodal* and *pool* functions, respectively, and assigned to the kth neuron of lth layer. In a heterogenous ONN configuration, every neuron has uniquely assigned ψ and P operators. Owing to this, an ONN network enjoys the flexibility of incorporating any non-linear transformation, which is suitable for the given learning problem. However, hand-crafting a suitable library of possible operators and searching for an optimal one for each neuron in a network introduces a significant overhead, which rises exponentially with increasing network complexity. Moreover, it is also possible that the ideal operator for the given learning problem cannot be expressed in terms of well-known functions. To resolve this key limitation, a composite nodal function is required that is iteratively created and tuned during back-propagation. A straightforward choice for accomplishing this would be to use a weighted combination of all operators in the operator set library and learn the weights during training. However, such a formulation would be susceptible to instability issues because of the different dynamic ranges of individual functions. Additionally, it would still rely on the manual selection of suitable functions to populate the operator set library. Therefore, to formulate a nodal transformation that does not require any pre-selection and manual assignment of operators, we make use of the Taylor-series based function approximation.

The Taylor series expansion of an infinitely differentiable function $f(x)$ near a point $x = a$ is given as:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n \qquad (5)$$

The Qth order truncated approximation of (5), formally known as the Taylor polynomial, takes the form of the following finite summation:

$$f(x)^{(Q)} = \sum_{n=0}^{Q} \frac{f^{(n)}(a)}{n!} (x-a)^n \qquad (6)$$

The above formulation enables the approximation of any function $f(x)$ sufficiently well in the close vicinity of a. If the coefficients $\frac{f^{(n)}(a)}{n!}$ are tuned and the inputs are bounded, the formulation of (6) can be used to *generate* any transformation. This is the key idea behind the generative neurons which form Self-ONNs. Specifically, in terms of the notation used in (4), the nodal transformation of a generative neuron would take the following general form:

$$\widetilde{\psi_k^l} \left(w_{ik}^{l(Q)}(r), y_i^{l-1}(m+r) \right)$$
$$= \sum_{q=1}^{Q} w_{ik}^{l(Q)}(r,q) \left(y_i^{l-1}(m+r) \right)^q \qquad (7)$$

In (7), Q is a hyperparameter which controls the degree of the Taylor series approximation, and $w_{ik}^{l(Q)}$ is a learnable kernel of the network. A key difference in (7) as compared to the convolutional (3) and operational (4) model is that $\widetilde{\psi_k^l}$ is not fixed, rather a distinct operator over each individual output, y_i^{l-1}, and thus requires Q times more parameters. Therefore, the $K \times 1$ kernel vector w_{ik}^l has been replaced by a $K \times Q$ matrix $w_{ik}^{l(Q)} \in \mathbb{R}^{K \times Q}$ which is formed by replacing each element $w_{ik}^l(r)$ with a Q-dimensional vector $w_{ik}^{l(Q)}(r) = [w_{ik}^{l(Q)}(r,0), w_{ik}^{l(Q)}(r,1), \dots, w_{ik}^{l(Q)}(Q-1)]$. The input map of the generative neuron, $\tilde{x}_{ik}^l$ can now be expressed as,

$$\widetilde{x_{ik}^l}(m) = P_k^l \left(\sum_{q=1}^{Q} w_{ik}^{l(Q)}(r,q) \left(y_i^{l-1}(m+r) \right)^q \right)_{r=0}^{K-1} \qquad (8)$$

During training, as $w_{ik}^{l(Q)}$ is iteratively tuned by the back-propagation (BP), *customized* nodal transformation functions will be generated as a result of (8), which would be uniquely tailored for ik^{th} connection. This enables an enhanced flexibility which provides three key benefits. Firstly, the need for manually defining a list of suitable nodal operators and searching for the optimal operator for each neuron connection is naturally alleviated. Secondly, the heterogeneity is not limited to each neuron connection $i \to k$ but down to each kernel element as $\tilde{\psi}_l^k \left(w_{ik}^{l(Q)}(r), y_i^{l-1}(m+r) \right)$ will be unique $\forall r \in [0,1,\dots,K-1]$. As illustrated in Figure 2, such a diversity is not achievable even with the flexible operational neuron model of ONNs. Thirdly, in generative neurons, the heterogeneity is driven only by the values of the weights $w_{ik}^{l(Q)}$ and the core operations (multiplication, summation) are the same for all neurons in a layer, as shown in (8). Owing to this, unlike ONNs, the generative neurons inside a Self-ONN layer can be parallelized much more efficiently, which leads to a considerable reduction in computational complexity and time. Moreover, a special case of (8) can also be expressed in terms of the widely applicable convolutional model.

A. Representation in terms of convolution

If the pooling operator P_k^l is fixed to summation operator, $\tilde{x}_{ik}^l$ is then defined as:

$$\widetilde{x_{ik}^l}(m) = \sum_{r=0}^{K-1} \sum_{q=1}^{Q} w_{ik}^{l(Q)}(r,q) \left(y_i^{l-1}(m+r) \right)^q \qquad (9)$$

Exploiting the commutativity of the summation operations in (9), we can alternatively write:

$$\widetilde{x_{ik}^l}(m) = \sum_{q=1}^{Q} \sum_{r=0}^{K-1} w_{ik}^{l(Q)}(r, q-1) y_i^{l-1}(m+r)^q \quad (10)$$

Using (1) and (2), the formula in (10) can be further simplified as follows:

$$\widetilde{x_{ik}^l} = \sum_{q=1}^{Q} Conv1D\left(w_{ik}^{l(Q)}, \left(y_i^{l-1}\right)^q\right) \quad (11)$$

Hence, the formulation can be accomplished by applying Q 1D convolution operations. If Q is set to 1, (11) entails the convolutional formulation of (3). Therefore, as CNN is a subset of ONN corresponding to a specific operator set, it is also a special case of Self-ONN with $Q = 1$ for all neurons.

B. Vectorized Notation

Expressing explicit loops in terms of matrix and vector manipulations is a key idea behind vectorization, which is a major driving factor behind fast implementations of modern-day neural network implementations. In this section, we first introduce how the vectorized notation can be used to express the 1D convolution operation inside a neuron. Afterward, the same key principles will be exploited to express the generative neuron formulation of (9) as a single matrix-vector product.
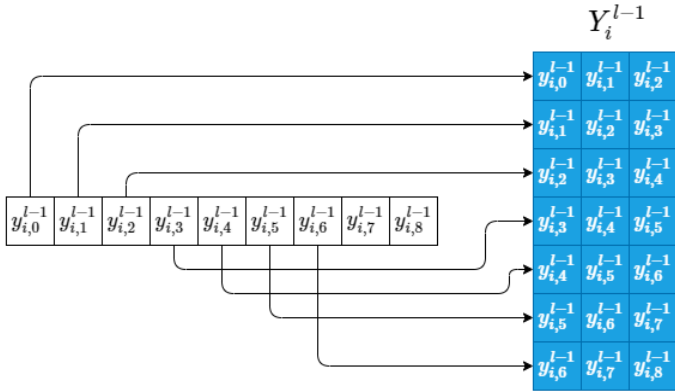
Figure 3: Reshuffling operation used to convert y_i^{l-1} to Y_i^{l-1}.

First, an alternate formulation of the operation of (3) is now presented. We introduce a transformation $\delta(\cdot, K)$ which concatenates y_i^{l-1} such that values inside each K-dimensional kernel as rows to form a matrix $Y_i^{l-1} \in \mathbb{R}^{M \times K}$. The process is visually depicted in Figure 3 for $K = 3$, and mathematically expressed in (12).

$$Y_i^{l-1} = \delta(y_i^{l-1}, K)$$
$$= \begin{bmatrix} y_i^{l-1}(0) & y_i^{l-1}(1) & \cdots & y_i^{l-1}(K-1) \\ \vdots & \vdots & \cdots & \vdots \\ y_i^{l-1}(m) & y_i^{l-1}(m+1) & \cdots & y_i^{l-1}(m+K-1) \\ \vdots & \vdots & \cdots & \vdots \\ y_i^{l-1}(M-1) & y_i^{l-1}(M) & \cdots & y_i^{l-1}(M+K-1) \end{bmatrix} \quad (12)$$

Secondly, we construct a matrix $W_{ik}^l \in \mathbb{R}^{M \times K}$ whose rows are repeated copies of $w_{ik} \in \mathbb{R}^K$.

$$W_{ik}^l = \begin{bmatrix} w_{ik}^l(0) & w_{ik}^l(1) & \cdots & w_{ik}^l(K-1) \\ \vdots & \vdots & \cdots & \vdots \\ w_{ik}^l(0) & w_{ik}^l(1) & \cdots & w_{ik}^l(K-1) \\ \vdots & \vdots & \cdots & \vdots \\ w_{ik}^l(0) & w_{ik}^l(1) & \cdots & w_{ik}^l(K-1) \end{bmatrix} \quad (13)$$

We now consider the Hadamard product of these two matrices:

$$Y_i^{l-1} \otimes W_{ik}^l$$
$$= \begin{bmatrix} y_i^{l-1}(0)w_{ik}^l(0) & \cdots & y_i^{l-1}(K-1)w_{ik}^l(K-1) \\ \vdots & \cdots & \vdots \\ y_i^{l-1}(m)w_{ik}^l(0) & \cdots & y_i^{l-1}(m+K-1)w_{ik}^l(K-1) \\ \vdots & \cdots & \vdots \\ y_i^{l-1}(M-1)w_{ik}^l(0) & \cdots & y_i^{l-1}(M+K-1)w_{ik}^l(K-1) \end{bmatrix} \quad (14)$$

Applying the summation operation across rows, we get:

$$\sum \left(Y_i^{l-1} \otimes W_{ik}^l\right)(m) = \sum_{r=0}^{K-1} w_{ik}^l(r) y_i^{l-1}(m+r) \quad (15)$$

which is equivalent to (3). We also note that,

$$\sum Y_i^{l-1} \otimes W_{ik}^l = Y_i^{l-1} w_{ik}^l \quad (16)$$

Therefore,

$$x_{ik}^l(m) = \left(Y_i^{l-1} w_{ik}^l\right)(m) \quad (17)$$

$$x_{ik}^l = Y_i^{l-1} w_{ik}^l \quad (18)$$

Hence, the 1D convolution operation can be represented in terms of a single matrix-vector product. This operation lies at the heart of conventional explicit general matrix multiplications (GEMM) based convolution implementations and enables efficient usage of parallel computational resources such as GPU cores.

C. Forward propagation through a 1D Self-ONN neuron

Eq. (11) shows how the Self-ONN formulation of (10) can be represented as a summation of Q individual convolutional operations. Moreover, from (12), a convolutional operation can be represented as a matrix-vector product. We now use these two formulations to represent the transformation of (11) as a single convolution operation, and consequently a single matrix-vector product, instead of Q-separate ones.

We start by introducing $Y_i^{l-1(Q)} \in \mathbb{R}^{M \times KQ}$ such that

$$Y_i^{l-1(Q)} = \left[Y_i^{l-1} \quad \left(Y_i^{l-1}\right)^{\circ 2} \quad \cdots \quad \left(Y_i^{l-1}\right)^{\circ Q} \right] \quad (19)$$

where $\circ n$ is the Hadamard exponentiation operator. The mth row of $Y_i^{l-1(Q)}$ can be expressed as,

$$Y_i^{l-1^{(Q)}}(m) = \begin{bmatrix} y_i^{l-1}(m) \\ \vdots \\ y_i^{l-1}(m+K-1) \\ \vdots \\ y_i^{l-1}(m)^2 \\ \vdots \\ y_i^{l-1}(m+K-1)^2 \\ \vdots \\ y_i^{l-1}(m)^Q \\ \vdots \\ y_i^{l-1}(m+K-1)^Q \end{bmatrix}^T \tag{20}$$

Moreover, we construct $W_{ik}^{l(Q)} \in \mathbb{R}^{M \times KQ}$ by first vectorizing $w_{ik}^{l(Q)} \in \mathbb{R}^{K \times Q}$ to $\overrightarrow{w_{ik}^{l(Q)}} \in \mathbb{R}^{KQ}$ and then concatenating m copies of $\overrightarrow{w_{ik}^{l(Q)}}$ along the row dimension, as expressed in (21) and (22).

$$\overrightarrow{w_{ik}^{l(Q)}} = \begin{bmatrix} w_{ik}^{l(Q)}(0,0) \\ \vdots \\ w_{ik}^{l(Q)}(K-1,0) \\ w_{ik}^{l(Q)}(0,1) \\ \vdots \\ w_{ik}^{l(Q)}(K-1,1) \\ \vdots \\ w_{ik}^{l(Q)}(0,Q-1) \\ \vdots \\ w_{ik}^{l(Q)}(K-1,Q-1) \end{bmatrix}^T \tag{21}$$

$$W_{ik}^{l(Q)}(m) = \overrightarrow{w_{ik}^{l(Q)}} \tag{22}$$

Taking the Hadamard product of $Y_i^{l-1^{(Q)}}$ and $W_{ik}^{l(Q)}$, we get:

$$\left(Y_i^{l-1^{(Q)}} \otimes W_{ik}^{l(Q)}\right)(m)$$

$$= \begin{bmatrix} y_i^{l-1}(m)w_{ik}^{l(Q)}(0,0) \\ \vdots \\ y_i^{l-1}(m+K-1)w_{ik}^{l(Q)}(K-1,0) \\ \vdots \\ y_i^{l-1}(m)^2 w_{ik}^{l(Q)}(0,1) \\ \vdots \\ y_i^{l-1}(m+K-1)^2 w_{ik}^{l(Q)}(K-1,1) \\ \vdots \\ y_i^{l-1}(m)^Q w_{ik}^{l(Q)}(0,Q-1) \\ \vdots \\ y_i^{l-1}(m+K-1)^Q w_{ik}^{l(Q)}(K-1,Q-1) \end{bmatrix}^T \tag{23}$$

Summation of the above yields:

$$\left(\sum \left(Y_i^{l-1^{(Q)}} \otimes W_{ik}^{l(Q)}\right)\right)(m)$$

$$= \sum_{r=0}^{K-1} y_i^{l-1}(m+r)w_{ik}^{l(Q)}(r,0)$$

$$+ \sum_{r=0}^{K-1} y_i^{l-1}(m+r)^2 w_{ik}^{l(Q)}(r,0)$$

$$+ \dots$$

$$+ \sum_{r=0}^{K-1} y_i^{l-1}(m+r)^Q w_{ik}^{l(Q)}(r,Q-1) \tag{24}$$

$$= \sum_{q=1}^{Q}\sum_{r=0}^{K-1} y_i^{l-1}(m+r)^q w_{ik}^{l(Q)}(r,q-1)$$

This is equivalent to (10). So, one can now express,

$$\left(\sum \left(Y_i^{l-1^{(Q)}} \otimes W_{ik}^{l(Q)}\right)\right)(m) = \widetilde{x_{ik}^l}(m) \tag{25}$$

Also, using (24), we can write,

$$\sum \left(Y_i^{l-1^{(Q)}} \otimes W_{ik}^{l(Q)}\right)$$

$$= \begin{bmatrix} \sum_{q=1}^{Q}\sum_{r=0}^{K-1} y_i^{l-1}(r)^q w_{ik}^{l(Q)}(r,q-1) \\ \vdots \\ \sum_{q=1}^{Q}\sum_{r=0}^{K-1} y_i^{l-1}(m+r)^q w_{ik}^{l(Q)}(r,q-1) \\ \vdots \\ \sum_{q=1}^{Q}\sum_{r=0}^{K-1} y_i^{l-1}(M-1+r)^q w_{ik}^{l(Q)}(r,q-1) \end{bmatrix} \tag{26}$$

Finally, from (25) and (26), we can simply infer that:

$$\widetilde{x_{ik}^l} = Y_i^{l-1^{(Q)}}\left(\overrightarrow{w_{ik}^{l(Q)}}\right) \tag{27}$$

The formulation of (27) provides a key computational benefit, as the forward propagation through the generative neuron is accomplished using a single matrix-vector multiplication. Hence, in theory, if the computational cost and memory requirement of constructing matrices $Y_i^{l-1^{(Q)}}$ and $W_{ik}^{l(Q)}$ is considered negligible, the complexity of a convolutional neuron is approximately the same as that of the generative neuron, as both can be accomplished by a single matrix-vector product. Finally, to complete the forward propagation, using (1), we can express:

$$\widetilde{x_k^l} = b_k^l + \sum_{i=0}^{N_{l-1}} \widetilde{x_{ik}^l} \tag{28}$$

D. Back-propagation

We now proceed to derive the back-propagation formulation for the generative neuron model of 1D Self-ONN by utilizing the vectorized notation introduced in Section II.C. To back-propagate the error through the generative neuron, given the derivative of the loss with respect to the neuron's output, $dL/d\widetilde{x_{ik}^{l}}$, we aim to define dL/dy_i^{l-1}, $dL/dw_{ik}^{l(Q)}$ and dL/db_k^l.

We start by taking the derivative of (27) w.r.t $\overline{Y_i^{l-1(Q)}}$ as follows:

$$\frac{d\widetilde{x_{ik}^{l}}(m)}{dY_i^{l-1(Q)}(\bar{m})} = \begin{cases} \overrightarrow{w_{ik}^{l(Q)}} & m = \bar{m} \\ \vec{0} & otherwise \end{cases} \tag{29}$$

Using (29), we can now apply the chain rule to get:

$$\frac{dL}{dY_i^{l-1(Q)}} = \frac{dL}{d\widetilde{x_{ik}^{l}}}\frac{d\widetilde{x_{ik}^{l}}}{dY_i^{l-1(Q)}} \tag{30}$$

Given $\frac{dL}{dY_i^{l-1(Q)}}$, we aim to find the derivative of the loss w.r.t to the previous layer's output:

$$\frac{dL}{dy_i^{l-1}} = \frac{dL}{dY_i^{l-1}}\frac{dY_i^{l-1}}{dy_i^{l-1}} \tag{31}$$

We know from (19) that $Y_i^{l-1(Q)} = \begin{bmatrix} Y_i^{l-1} & \left(Y_i^{l-1}\right)^{\circ 2} & \cdots & \left(Y_i^{l-1}\right)^{\circ Q} \end{bmatrix}$. Taking the derivative of (19) w.r.t Y_i^{l-1}:

$$\frac{dY_i^{l-1(Q)}}{dY_i^{l-1}} = \begin{bmatrix} 1 & 2\left(Y_i^{l-1}\right)^{\circ 1} & \cdots & Q\left(Y_i^{l-1}\right)^{\circ Q-1} \end{bmatrix} \tag{32}$$

Using this we can write:

$$\frac{dL}{dY_i^{l-1}} = \frac{dL}{dY_i^{l-1(Q)}}\frac{dY_i^{l-1(Q)}}{dY_i^{l-1}} \tag{33}$$

Finally, we can calculate the derivative of the loss w.r.t y_i^{l-1} as follows:

$$\frac{dL}{dy_i^{l-1}(\bar{m})} = \sum_{m=0}^{M-1}\frac{dL}{dY_i^{l-1}(m)}\frac{dY_i^{l-1}(m)}{dy_i^{l-1}(\bar{m})}$$
$$= \sum_{m=0}^{M-1}\frac{dL}{dY_i^{l-1}(m)}\left[\frac{dy_i^{l-1}(m)}{dy_i^{l-1}(\bar{m})},\ldots,\frac{dy_i^{l-1}(m+K-1)}{dy_i^{l-1}(\bar{m})}\right] \tag{34}$$

From (34) and (12), we can notice that $\left(\frac{dY_i^{l-1}(m)}{dy_i^{l-1}(\bar{m})}\right)$ will be equal to 1 only when the condition $m \le \bar{m} \le (m+K-1)$ is met, and 0, otherwise. Moreover, as there are no repeating entries in each row of Y_i^{l-1}, only one element of $\frac{dY_i^{l-1}(m)}{dy_i^{l-1}(\bar{m})}$ can be non-zero and the location of this non-zero element is given

by $mod(\bar{m}, K)$. Based on these two observations, we can infer the following:

$$\frac{dL}{dy_i^{l-1}(\bar{m})}$$
$$= \sum_{m=0}^{M-1}\begin{cases}\frac{dL}{dY_i^{l-1}}(m, mod(\bar{m}, K)) & m \le \bar{m} \le (m+K \\ 0, & otherwise \end{cases} \tag{35}$$

The only other partial derivative needed for completing the back-propagation is the of the loss w.r.t the weights of the neuron $\overrightarrow{w_{ik}^{l(Q)}}$. Again, by the chain rule, we can write:

$$\frac{dL}{dw_{ik}^{l(Q)}}(\bar{r}) = \frac{dL}{d\widetilde{x_{ik}^{l}}}\frac{d\widetilde{x_{ik}^{l}}}{dw_{ik}^{l(Q)}} \tag{36}$$

where $\frac{d\widetilde{x_{ik}^{l}}}{dw_{ik}^{l(Q)}}$ can be calculated by taking the derivative of (27) w.r.t $\overrightarrow{w_{ik}^{l(Q)}}$ as follows:

$$\frac{d\widetilde{x_{ik}^{l}}}{dw_{ik}^{l(Q)}} = Y_i^{l-1(Q)} \tag{37}$$

For the bias, we can use (28) to write:

$$\frac{dL}{db_k^l} = \frac{dL}{d\widetilde{x_k^l}}\frac{d\widetilde{x_k^l}}{db_k^l} = \sum_{m=0}^{M-1}\frac{dL}{d\widetilde{x_k^l}}(m) \tag{38}$$

Finally, assuming a stochastic gradient descent (SGD) based optimization, the weights and biases can be updated as follows:

$$\overrightarrow{w_{ik}^{l(Q)}}(t+1) = \overrightarrow{w_{ik}^{l(Q)}}(t) - \epsilon(t)\frac{dL}{dw_{ik}^{l(Q)}} \tag{39}$$

$$b_k^l(t+1) = b_k^l(t) - \epsilon(t)\frac{dL}{db_k^l} \tag{40}$$

where $\epsilon(t)$ is the learning factor at iteration t.

III. METHODOLOGY

In this study, we are aiming to achieve a superior R-peak detection performance compared to the state-of-the-art method in [34] based on a 12-layer 1D CNN with 448 neurons while reducing the network complexity and depth significantly. Therefore, as illustrated in Figure 4, for a fair comparison with [34], the same R-peak approach is followed. However, the architectural complexity of the network is reduced by using a 6-layer 1D Self-ONN with less than 100 neurons. In order to perform fair comparative evaluations against [34], the same

network model of [34], the UNet is used for Self-ONNs. As shown in the figure, the peak detection problem is converted to a regression task, which aims to learn the peak locations by transforming the original (normalized) ECG segment into a sequence of 5-sample wide pulses. The center of each pulse corresponds to the R-peak location. In order to evaluate its effect, the order of the Taylor polynomials, Q, is varied in {3,5,7} over three networks. Each 20-seconds (8000 samples)

ECG segment is linearly normalized in the range, [-1, 1], and then used as input for the 1D Self-ONNs. The same optimizer (Adam) is used to train the network with the same number of epochs and hyperparameters. The details of the experimental setup and network parameters will be presented in the next section.
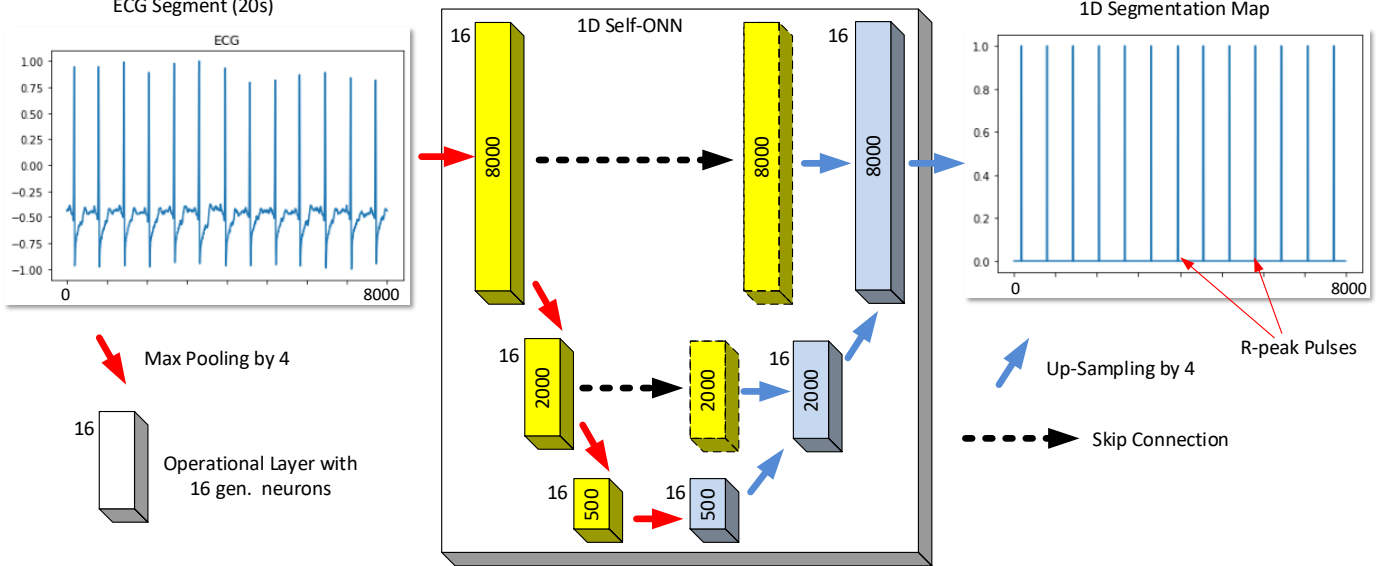

Figure 4: The proposed approach for R-peak detection.

IV. EXPERIMENTAL RESULTS

In this section, we will first introduce the benchmark ECG dataset, China Physiological Signal Challenge-2020 (CPSC), used in this study, and then present the experimental setup used for testing and evaluation of the proposed R-peak detector using 1D Self-ONNs. An extensive set of ECG classification experiments and comparative evaluations against the recent methods over the benchmark CPSC-DB Holter dataset will be presented next. Finally, the computational complexity analysis of the 1D Self-ONNs and 1D CNNs will be reported in detail.

A. China Physiological Signal Challenge-2020

The China Physiological Signal Challenge (2020) dataset (CPSC-DB) consists of 10 single-lead ECG recordings which are collected from arrhythmia patients, each of the recordings lasts for about 24 hours. Table I presents the patient information in detail [41].

Table I: Patient information on the ECG data from CPSC-DB.

Patient	AF?	Length(h)	# Tot. Beats	# V Beats	# S Beats
1	No	25.89	109731	0	24
2	Yes	22.83	108297	4554	0
3	Yes	24.70	138878	382	0
4	No	24.51	101734	19024	3466
5	No	23.57	94635	1	25
6	No	24.59	770806	0	6
7	No	23.11	96814	15150	3481
8	Yes	25.46	125495	2793	0
9	No	25.84	89854	2	1462
10	No	23.64	82851	169	9071

The other properties of the CPSC-DB (Holter) ECG dataset can be summarized as follows: All ECG data were acquired by a unified wearable ECG device with a sampling frequency of 400 Hz and the total number of beats is 1,026,095. The recordings include irregular heart rhythms as well as supraventricular premature beats (SPBs or S beats) and premature ventricular contraction (PVC or V) type beats. All recordings are provided in MATLAB format with corresponding S and V beats annotations. R-peak annotations for each ECG cycle were annotated by a team of biomedical researchers. To show the robustness of the R-peak detector against noise and other artifacts, CPSC-DB presents a real-world Holter dataset containing numerous ECG containments and artifacts.

B. Experimental Setup

For all experiments, as in [34] a shallow training is employed where the number of back-propagation (BP) epochs is limited to 50. We set the learning rate, $\epsilon(0)$, as 10^{-3} and Adam optimizer is used for minimizing the binary cross-entropy loss (BCE). We performed three individual BP runs for training over each patient's data and for comparative evaluations, we report the best detection performance.

The comparative evaluations of the proposed R-peak detector based on 1D Self-ONNs are carried out against the following two detectors with 1D CNNs: i) the deep 1D CNN from [34], and ii) 1D CNN with the same network configuration as 1D Self-ONN. Moreover, comparative evaluations are also performed against five earlier *state-of-the-art* classifiers from the literature, [23], [56]-[59]. R-peak detection experiments are performed over the CSPC dataset. Each classifier is trained

over 9 patients and tested over the (unseen) patient. Therefore, 10-fold cross-validation is performed for comparative evaluations over all 10 patients with overall 1,026,095 beats. Over the cumulated true-positive (TP), false-positive (FP) and false-negative (FN) counters, and the standard performance metrics, *Precision* or *Positive Predictivity* (*Ppr*), *Recall* or *Sensitivity* (*Sen*), and *F1-score*, which is the harmonic mean of the model's *Precision* and *Recall*. are computed. The true-negative (TN) is omitted in peak detection (TN=0). The expressions of these performance metrics using the hit/miss counters, e.g., true positive (*TP*), true negative (*TN*), false positive (*FP*), and false negative (*FN*), are as follows:

$$Ppr = \frac{TP}{TP + FP}, Sen = \frac{TP}{TP + FN}, F1 = \frac{2PprSen}{Ppr + Sen} \quad (41)$$

C. Peak Detection Performance Evaluation

R-peak detection results over the entire CSPC dataset are presented in Table II. The (performance) loss (%) in FN (or FP or other "miss" metrics) of a method *X* compared to the method with the minimum FN can be defined as follows:

$$FN\ LOSS(X)\% = 100\frac{FN(min)}{FN(X)} \quad (42)$$

It is clear that 1D Self-ONNs achieved a significant performance gap when compared to the 1D CNNs with the same number of learning units (neurons) and depth. Overall, Self-ONNs with Q=3 and Q=5 reduced the FPs and FNs on peak detection by more than 43% and 64%, respectively. Even when compared with the state-of-the-art 1D CNN configuration in [34] with twice the depth and more than 4 times the number of neurons, the detection errors, FPs and FNs, were reduced by more than 7% and 37%, respectively. Such a substantial reduction especially on FNs over both 1D CNN configurations shows that 1D Self-ONNs can indeed accomplish a superior learning capability to detect the actual peaks. The performance loss is significantly higher for the earlier methods [23], [56]-[59]. Finally, the results indicate that the best performances are obtained by 1D Self-ONNs with either *Q*=3 or *Q*=5. We have found that the former setting is a better choice because of the higher F1 score, significantly lower FPs, and lower computational cost.

Table II: Overall peak detection performance of the classifiers. 1D CNN (*) and 1D Self-ONN have the same configuration. The best results are presented in **bold**.

Network	Q	TP	FN	FN LOSS (%)	FP	FP LOSS (%)	Sen	Ppr	F1
LSTM [56]	-	1,007,823	18,272	89.02	23,835	45.88	98.20	97.76	97.88
P and T [23]	-	998,413	27,682	92.75	28,940	55.43	97.31	97.18	97.23
Hamilton [57]	-	993,920	32,175	93.76	62,733	79.44	96.82	93.73	95.14
Two M. Avg. [58]	-	992,305	33,790	94.06	46,349	72.17	96.66	95.34	95.97
SWT [59]	-	975,222	50,873	96.05	27,936	53.83	95.06	97.20	96.09
1D CNN [34]	1	1,022,874	3221	37.69	13931	7.41	99.70	98.63	99.16
1D CNN (*)	1	1,020,544	5651	64.48	22820	43.48	99.43	97.83	98.57
1D Self-ONN	3	1,023,997	2098	4.34	**12899**	**0.00**	99.80	**98.77**	**99.28**
	5	**1,024,088**	**2007**	**0.00**	14263	9.56	**99.81**	98.63	99.21
	7	1,023,907	2188	8.27	16481	21.73	99.79	98.42	99.10

Table III: The peak detection performance of the classifiers over the arrhythmia (S and V) beats. 1D CNN (*) and 1D Self-ONN have the same configuration. The best results are presented in **bold**.

Network	Q	S beats			V beats		
		Detected	Miss	LOSS (%)	Detected	Miss	LOSS (%)
LSTM [56]	-	17,330	205	90.73	29,247	12,828	98.21
P and T [23]	-	17,230	305	93.77	31,369	10,706	97.85
Hamilton [57]	-	17,163	372	94.89	31,906	10,169	97.74
Two M. Avg. [58]	-	17,231	304	93.75	28,972	13,103	98.44
SWT [59]	-	17,018	517	96.32	27,283	14,792	
1D CNN [34]	1	17,495	40	52.50	41,523	552	58.33
1D CNN (*)	1	17,463	72	73.61	41,687	738	68.83
1D Self-ONN	3	17,504	31	38.71	**41,805**	**230**	**0.00**
	5	**17,516**	**19**	**0.00**	41,755	235	2.13
	7	17,507	28	32.14	41,713	287	19.86

In peak detection, detecting the arrhythmia, i.e., Supraventricular Premature (S) and Premature Ventricular Contraction (V) beats is crucial since R-peak detection is the prior operation to an automated ECG beat classification and arrhythmia detection. Obviously, this aim cannot be fulfilled if the peak detector fails to detect an abnormal beat. For this

purpose, the focus is then drawn on the peak detection performance over the arrhythmia (S and V) beats. As shown in Table III, once again 1D Self-ONNs have shown superiority for detecting both S and V beats over both 1D CNN configurations. This time even the deep 1D CNN causes over 52% and 58% more misdetections, respectively, and again, the gap further widens over the 1D CNN with the same configuration. Finally, the loss of the earlier methods [23], [56]- [59] has peaked above 90%, which shows that those methods are not robust at all for Holter ECG.

As for visual comparison, over the four ECG segments, Figure 5 shows R-peak detection results of 1D Self-ONNs and the two 1D CNN configurations. Such typical visual results clearly show that both 1D CNN models yield numerous FPs and FNs especially when the R-peak is corrupted with high noise or some abrupt change occurring in the close vicinity, e.g., abrupt shifts on the baseline or occasional voltage glitches. Especially, the deep CNN model [34] missed both arrhythmic peak locations in plot (d) which is a substantial error.
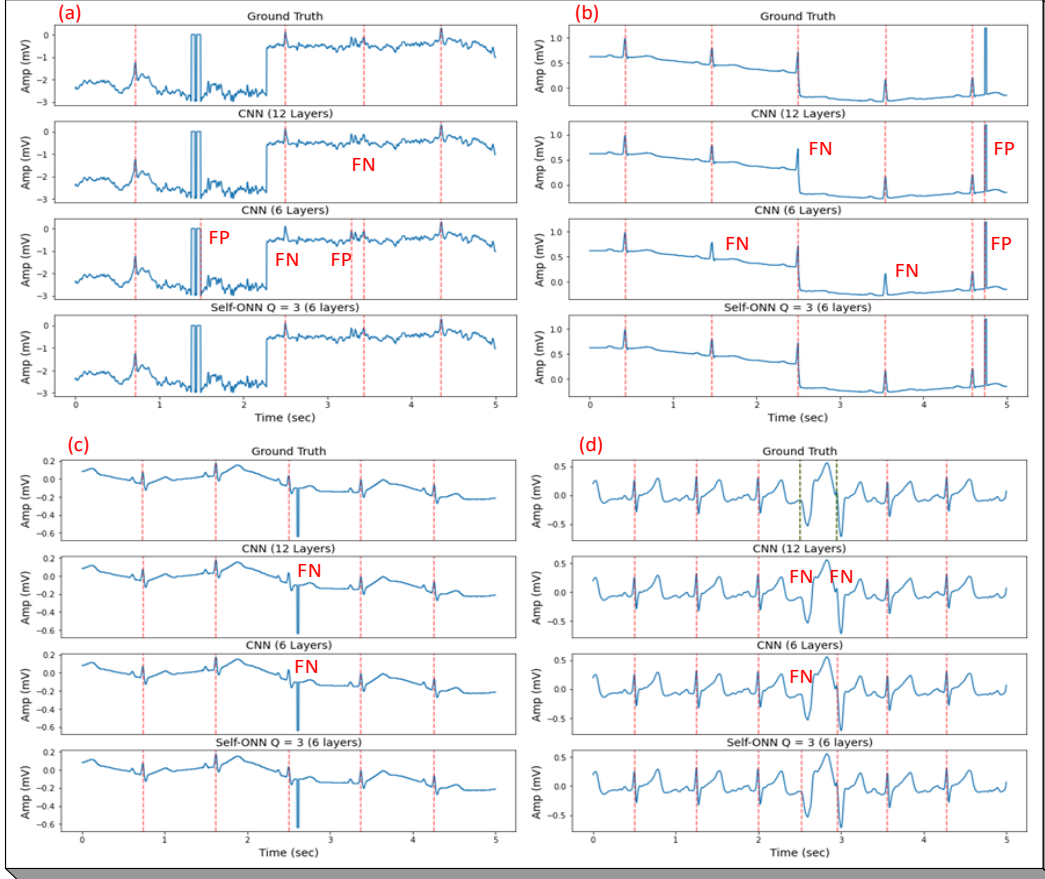

Figure 5: Four R-peak detection results of 1D Self-ONNs (bottom of each plot) and the two 1D CNN configurations ([34] in the 2nd row and the 6-layer 1D CNN with the same configuration is in the 3rd row). Typical false-positives (FPs) and false-negatives (FNs) are visible over the detections of both 1D CNN models. Ground truth peak locations are shown on the top plot where red and black dashed lines correspond to normal and arrhythmic peak locations, respectively.

D. Computational Complexity

In this section, we provide the formulation for calculating the total number of multiply-accumulate operations (MACs) and the total number of parameters (PARs) of a generative neuron of a 1D Self-ONN. To calculate the number of trainable parameters, we recall from Section II that, for each kernel connection, the generative neuron has Q times more learnable parameters. Cumulatively, the number of trainable parameters, n_k^l, of the kth neuron of lth layer, is given by the following formulation:

$$n_k^l = N_{l-1} * K_k^l * Q_k^l \qquad (43)$$

In (43), N_{l-1} is the number of neurons in layer $l-1$, K_k^l is the kernel size used in the neuron and Q_k^l is the approximation order selected for this neuron. Finally, to calculate the total number of MAC operations, one can note from (26) that to produce a single element in the output $\widetilde{x_{ik}^l}$, we require $K_k^l * Q_k^l$ MAC operations for each output map y_i^{l-1} of the previous layer. Generalizing this, we can write the following:

$$MAC_k^l = N_{l-1} * \left| \widetilde{x_{ik}^l} \right| * K_k^l * Q_k^l \qquad (44)$$

where $|\cdot|$ is the cardinality operator. For notational convenience, the bias term and the cost of Hadamard exponentiation are omitted from (44).

Table IV: Network models and their computational complexities. The average time corresponds to the time to detect R peak locations of a 20s ECG segment.

Network	PARs (K)	MACs (M)	Avg. Time (ms) CPU	Avg. Time (ms) GPU	Train Time (min)
1D CNN [34]	79,121	67.05	191	110	37.0
1D CNN (*)	12,801	46.36	90	64	11.8
1D Self-ONN (Q=3)	38,209	138.65	106	66	19.5

We implemented the proposed 1D Self-ONN using Python and FastONN [61] library, based on PyTorch [62]. All the experiments reported in this paper were run on a 2.2GHz Intel Core i7-8750H with 16 GB of RAM and NVIDIA GeForce GTX 1060 graphic card. Both training and evaluation of the classifier were processed by CUDA kernels. Along with the average time complexity, using the formulations in (43) and (44), we provide the overall PARs and MACs for both network models in Table IV. As shown in the table, a significant gap occurs between 1D Self-ONN and deep CNN models in terms of the total number of parameters and average computation time. The Self-ONN network requires the most number of multiply-accumulate operations. However, a higher percentage of these operations are independent, and thus parallelizable. Therefore, an efficient implementation using the formulation of (27) results in an actual running time that is less than the deep CNN and only marginally (3.1%) higher than the equivalent 1D CNN, which is negligible considering the crucial gain in the detection performance. The optimized PyTorch implementation of Self-ONNs are publicly shared in [63].

V. CONCLUSIONS

In this study, 1D Self-ONNs are proposed for R-peak detection especially for poor quality ECG signals, e.g., acquired by Holter monitors or low-power mobile ECG sensors. The primary goal is to achieve the state-of-the-art R-peak detection performance with an elegant computational efficiency for a real-time application over such low-power ECG devices. As a new-generation network model, a Self-ONN is a highly heterogeneous network composed of generative neurons. This yields a crucial advantage of optimizing the nodal operator function of each kernel element and thus, Self-ONNs can achieve an utmost heterogeneity that maximizes the network diversity and the learning performance. As a result, the traditional weight optimization of conventional CNNs is entirely turned into an operator generation process via optimization. Despite its highly non-linear kernel elements, each Self-ONN layer can still be implemented by a single 1D convolution, and this allows a parallelized implementation similar to the one for conventional CNNs.

We performed 10-fold comparative evaluations over the benchmark CPSC dataset with more than 1M beats. Against the current *state-of-the-art* method proposed in [34] with a 12-layer CNN, 1D Self-ONNs significantly reduced both FPs and FNs even though it has *half* the depth and more than 4 times fewer neurons. Against the 1D CNN with the equivalent configuration, the performance gap further widens. The most crucial advantage is that 1D Self-ONNs can reduce more than 52% and 58% of the overall misdetections of the S and V arrhythmia beats, respectively, compared to the deep CNNs. Finally, the 1D Self-ONN model used in this study presents a superior computational efficiency with respect to the deep 1D CNNs, and thus, especially for low-power, mobile devices such as Holter monitors, the proposed approach can conveniently be used as an R-peak detector in real-time. The optimized implementation of the proposed peak detector and the benchmark CPSC dataset are shared in [64].

REFERENCES

[1] World Health Organization. Cardiovascular Diseases (CVDs) Fact Sheet. Accessed: May 20, 2018. [Online]. Available: http://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases.

[2] O.T. Inan, L. Giovangrandi, and G.T.A. Kovacs, "Robust neural-network based classification of PVCs using wavelet transform and timing interval features," IEEE Trans. Biomed. Eng., vol. 53, no. 12, pp. 2507–2515, Dec. 2006.

[3] X. Alfonso and T. Q. Nguyen, "ECG beat detection using filter banks," IEEE Trans. Biomed. Eng., vol. 46, no. 2, pp. 192–202, Feb. 1999.

[4] J. L. Willems and E. Lesaffre, "Comparison of multigroup logistic and linear discriminant ECG and VCG classification," J. Electrocardiol., vol. 20, pp. 83-92, 1987.

[5] J. L. Talmon, Pattern Recognition of the ECG. Berlin, Germany: Akademisch Proefscript, 1983.

[6] D. A. Coast, R. M. Stern, G. G. Cano, and S. A. Briller, "An approach to cardiac arrhythmia analysis using hidden Markov models," IEEE Trans. Biomed. Eng., vol. 37, no. 9, pp. 826–836, Sep. 1990.

[7] S. Osowski, L. T. Hoai, and T. Markiewicz, "Support vector machine based expert system for reliable heartbeat recognition," IEEE Trans.Biomed. Eng., vol. 51, no. 4, pp. 582–589, Apr. 2004.

[8] Y. H. Hu, W. J. Tompkins, J. L. Urrusti, and V. X. Afonso, "Applications of artificial neural networks for ECG signal detection and classification," J. Electrocardiol., pp. 66–73, 1994.

[9] Y. Hu, S. Palreddy, and W. J. Tompkins, "A patient-adaptable ECG beat classifier using a mixture of experts approach," IEEE Trans. Biomed. Eng., vol. 44, no. 9, pp. 891–900, Sep. 1997.

[10] S. C. Lee, "Using a translation-invariant neural network to diagnose heart arrhythmia," in IEEE Proc. Conf. Neural Information Processing Systems, Nov. 1989.

[11] P. de Chazal and R. B. Reilly, "A patient-adapting heartbeat classifier using ECG morphology and heartbeat interval features," IEEE Trans. Biomed. Eng., vol. 53, no. 12, pp. 2535–2543, Dec. 2006.

[12] U.R. Acharya, H. Fujitad, O. S. Liha, Y. Hagiwaraa, J. H. Tana, M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network," Inf. Sci. 405 (2017) 81–90.

[13] U.R. Acharya, et al., "A deep convolutional neural network model to classify heartbeats," Comput. Biol. Med. 89 (2017) 389–396.

[14] M. Zubair, J. Kim, C. Yoon, "An automated ECG beat classification system using convolutional neural networks," 2016 6th Int. Conf. on IT Convergence and Security (ICITCS), 2016.

[15] S.S. Xu, M. Mak, C. Cheung, "Towards end-to-end ECG classification with raw signal extraction and deep neural networks," IEEE J Biomed Health Inform., vol. 23(4), pp. 1574–1584, 2019.

[16] Y. Xia, Y. Xie, "A novel wearable electrocardiogram classification system using convolutional neural networks and active learning," IEEE Access, vol. 7, pp.7989–8001, 2019.

[17] T. Ince, S. Kiranyaz, and M. Gabbouj, "A Generic and Robust System for Automated Patient-specific Classification of Electrocardiogram Signals", IEEE Transactions on Biomedical Engineering, vol. 56, issue 5, pp. 1415-1426, May 2009.

[18] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1D Convolutional Neural Networks", *IEEE Trans. on Biomedical Engineering*, vol. 63, no. 3, pp. 664-675, Mar 2016.

[19] S. Kiranyaz, T. Ince, M. Gabbouj and R. Hamila, "Convolutional Neural Networks for Patient-Specific ECG Classification", the 37th *IEEE Engineering in Medicine and Biology Society Conference (EMBC'15)*. Milano, Italy, Aug. 2015.

[20] X. Xu, H. Liu, "Ecg heartbeat classification using convolutional neural networks.*" IEEE Access*, vol. 8, pp. 8614–8619, 2020.

[21] S. Kiranyaz, T. Ince, and M. Gabbouj, "Personalized Monitoring and Advance Warning System for Cardiac Arrhythmias," *Scientific Reports*, vol. 7, Aug. 2017. DOI 10.1038/s41598-017-09544-z (SREP-16-52549-T) (http://rdcu.be/vfYE).

[22] F. Liu *et al.*, "Performance Analysis of Ten Common QRS Detectors on Different ECG Application Cases," *J. Healthc. Eng.*, vol. 2018, 2018, doi: 10.1155/2018/9050812.

[23] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Trans. Biomed. Eng.*, vol. BME-32, no. 3, pp. 230–236, 1985, doi: 10.1109/TBME.1985.325532.

[24] 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, doi: 10.1109/ACCESS.2020.2967775.

[25] V. Kalidas and L. Tamil, "Real-time QRS detector using stationary wavelet transform for automated ECG analysis," 2017, doi: 10.1109/BIBE.2017.00-12. (journal/conf??)

[26] M. Rakshit and S. Das, "An efficient wavelet-based automated R-peaks detection method using Hilbert transform," Biocybern. Biomed. Eng., vol. 37, no. 3, pp. 566–577, 2017, doi: 10.1016/j.bbe.2017.02.002.

[27] B. Hossain, S. Khairul 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," doi: 10.1109/ACCESS.2019.2939943. (journal/conference??)

[28] K. Arbateni and A. Bennia, "Sigmoidal radial basis function ANN for QRS complex detection," *Neurocomputing*, vol. 145, pp. 438–450, Dec. 2014, doi: 10.1016/j.neucom.2014.05.009.

[29] A. S. A. Huque, K. I. Ahmed, M. A. Mukit, and R. Mostafa, "HMM-based Supervised Machine Learning Framework for the Detection of fECG R-R Peak Locations," IRBM, vol. 40, no. 3, pp. 157–166, Jun. 2019, doi: 10.1016/j.irbm.2019.04.004.

[30] Moody GB, Muldrow WE, and Mark RG, "The MIT-BIH Noise Stress Test Database," in Computers in Cardiology, 1984, pp. 381–384, doi: 10.13026/C2HS3T.

[31] B. Porr and L. Howell, "R-peak detector stress test with a new noisy ECG dataset reveals significant performance differences amongst popular detectors," bioRxiv, p. 722397, Aug. 2019, doi: 10.1101/722397.

[32] J. Laitala et al., "Robust ECG R-peak detection using LSTM," Proc. ACM Symp. Appl. Comput., no. March, pp. 1104–1111, 2020, doi: 10.1145/3341105.3373945.

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Lecture Notes in Computer Science, May 2015, vol. 9351, pp. 234–241, doi: 10.1007/978-3-319-24574-4_28.

[34] M. U. Zahid, S. Kiranyaz, O. C. Devecioglu, M. E. H. Chowdhury, A. Khandakar, A. Tahir, and M. Gabbouj, "Robust R-Peak Detection in Low-Quality Holter ECGs using 1D Convolutional Neural Network", arXiv preprint arXiv:2101.01666, Jan. 2021.

[35] R. Mark and G. Moody, "MIT-BIH Arrhythmia Dataset Directory," [Online]. Available: http://ecg.mit.edu/dbinfo.html

[36] "Recommended practice for testing and reporting performance results of ventricular arrhythmia detection algorithms," Association for the Advancement of Medical Instrumentation, Arlington, VA, 1987.

[37] P. de Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," IEEE Trans. Biomed. Eng., vol. 51, no. 7, pp. 1196–1206, Jul. 2004.

[38] W. Jiang and S. G. Kong, "Block-based neural networks for personalized ECG signal classification," IEEE Trans. Neural Networks, vol. 18, no. 6, pp. 1750–1761, Nov. 2007.

[39] M. Llamedo, J.P. Martinez, "An Automatic Patient-Adapted ECG Heartbeat Classifier Allowing Expert Assistance," *IEEE Transactions on* Self-ONN source [Online] http://selfonn.net/ .

[40] *Biomedical Engineering*, vol. 59, issue 8, pp. 2312-2320, Aug 2012.

[41] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Math. Biophysics, 5:115–133, 1943.

[42] T. Klausberger, P. Somogyi, "Neuronal diversity and temporal generatives: the unity of hippocampal circuit operations" , Science, 321, pp. 53-57, (5885), 2008.

[43] I. Soltesz, "Diversity in the Neuronal Machine: Order and Variability in Interneuronal Microcircuits", Oxford University Press, USA, 2002.

[44] S. Kiranyaz, T. Ince, A. Iosifidis and M. Gabbouj, "Generalized model of biological neural networks: progressive operational perceptrons," IJCNN, 2017.

[45] S. Kiranyaz, T. Ince, A. Iosifidis and M. Gabbouj, "Progressive operational perceptrons," *Neurocomputing*, vol. 224, pp.142–154, 2017.

[46] D.T. Tran, S. Kiranyaz, M. Gabbouj and A. Iosifidis, "Progressive operational perceptron with memory," *Neurocomputing,* 379, pp. 172-181, 2020.

[47] D.T. Tran and A. Iosifidis, "Learning to Rank: A progressive neural network learning approach", IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, Brighton, U.K., 2019.

[48] D.T. Tran, S. Kiranyaz, M. Gabbouj and A. Iosifidis, "Heterogeneous Multilayer Generalized Operational Perceptron", *IEEE Trans. on Neural Networks and Learning Systems*, pp. 1-15, May 2019. DOI: 10.1109/TNNLS.2019.2914082.

[49] D.T. Tran, S. Kiranyaz, M. Gabbouj and A. Iosifidis, "Knowledge Transfer for Face Verification Using Heterogeneous Generalized Operational Perceptrons," IEEE Int. Conf. on Image Processing, Taipei, Taiwan, 2019.

[50] S. Kiranyaz, T. Ince, A. Iosifidis and M. Gabbouj, "Operational Neural Networks", Neural Computing and Applications, Mar. 2020. DOI: https://doi.org/10.1007/s00521-020-04780-3

[51] S. Kiranyaz, J. Malik, H. B. Abdallah, T. Ince, A. Iosifidis, M. Gabbouj, "Self-Organized Operational Neural Networks with Generative Neurons", Neural Networks, vol. 140, pp. 294-308 https://doi.org/10.1016/j.neunet.2021.02.028 , 2020.

[52] T. Ince, J. Malik, O. Devecioglu, S. Kiranyaz, O. Avci, L. Eren, M. Gabbouj, "Early Bearing Fault Diagnosis of Rotating Machinery by 1D Self-Organized Operational Neural Networks", IEEE Access. vol. 9, pp. 139260-139270, 2021, doi: 10.1109/ACCESS.2021.3117603

[53] J. Malik, O. C. Devecioglu, S. Kiranyaz, T. Ince, M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1D Self-Operational Neural Networks," *IEEE Trans. on Biomedical Eng.,* Dec. 2021. DOI: 10.1109/TBME.2021.3135622

[54] J. Malik, S. Kiranyaz, M. Gabbouj, "Self-Organized Operational Neural Networks for Severe Image Restoration Problems", Neural Networks, 2021. DOI: https://doi.org/10.1016/j.neunet.2020.12.014

[55] O. C. Devecioglu, J. Malik, T. Ince, S. Kiranyaz, E. Atalay, M. Gabbouj, "Real-Time Glaucoma Detection From Digital Fundus Images Using Self-ONNs", *IEEE Access*, Vol. 9, pp. 140031-140041, Sep. 2021. DOI: 10.1109/ACCESS.2021.3118102

[56] J. Laitala *et al.*, "Robust ECG R-peak detection using LSTM," *Proc. ACM Symp. Appl. Comput.*, no. March, pp. 1104–1111, 2020. doi: 10.1145/3341105.3373945.

[57] P. S. Hamilton and E. P. Limited, "Open Source ECG Analysis Software Documentation," 2002. Accessed: Dec. 10, 2020. [Online]. Available: http://www.eplimited.com/ .

[58] M. Elgendi, M. Jonkman, and F. De Boer, "Frequency bands effects on QRS detection." Institute for Systems and Technologies of Information, Control and Communication (INSTICC), pp. 428–431, Jan. 2010.

[59] V. Kalidas and L. Tamil, "Real-time QRS detector using Stationary Wavelet Transform for Automated ECG Analysis," 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), Washington, DC, 2017, pp. 457-461, doi: 10.1109/BIBE.2017.00-12.

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

[61] J. Malik, S. Kiranyaz, M. Gabbouj, "FastONN – Python based open-source GPU implementation for Operational Neural Networks", arXiv: https://arxiv.org/abs/2006.02267 . 2020.

[62] A. Paszke, A., et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: H. Wallach, et al. (Eds.), Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8026–8037.

[63] Self-ONN source [Online] http://selfonn.net/ .

[64] 1D Self-ONN Peak Detector [Online] https://github.com/MUzairZahid/R-Peak-Detection-1D-SelfONN .