IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 July 2022, accepted 31 August 2022, date of publication 5 September 2022, date of current version 14 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204703

RESEARCH ARTICLE

A Self-Contained STFT CNN for ECG Classification and Arrhythmia Detection at the Edge

MOHAMMED M. FARAG, (Member, IEEE)
Electrical Engineering Department, College of Engineering, King Faisal University, Al Ahsa 31982, Saudi Arabia
Electrical Engineering Department, Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt

e-mail: mfarag@kfu.edu.sa

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia, under Project GRANT697.

ABSTRACT Automated classification of Electrocardiogram (ECG) for arrhythmia monitoring is the core of cardiovascular disease diagnosis. Machine Learning (ML) is widely used for arrhythmia detection. The cloud-based inference is the prevailing deployment model of modern ML algorithms which does not always meet the availability and privacy requirements of ECG monitoring. Edge inference is an emerging alternative that addresses the concerns of latency, privacy, connectivity, and availability. However, edge deployment of ML models is challenging due to the demanding requirements of modern ML algorithms and the computation constraints of edge devices. In this work, we propose a lightweight self-contained short-time Fourier Transform (STFT) Convolutional Neural Network (CNN) model for ECG classification and arrhythmia detection in real-time at the edge. We provide a clear interpretation of the convolutional layer as a Finite Impulse Response (FIR) filter and exploit this interpretation to develop an STFT-based 1D convolutional (Conv1D) layer to extract the spectrogram of the input ECG signal. The Conv1D output feature maps are reshaped into a 2D heatmap image and fed to a 2D convolutional (Conv2D) neural network (CNN) for classification. The MIT-BIH arrhythmia database is used for model training and testing. Four model variants are trained and tested on a cloud machine and then optimized for edge computing on a raspberry-pi device. Weight quantization and pruning techniques are applied to optimize the developed models for edge inference. The proposed classifier can achieve up to 99.1% classification accuracy and 95% F1-score at the edge with a maximum model size of 90 KB, an average inference time of 9 ms, and a maximum memory usage of 12 MB. The achieved results of the proposed classifier enable its deployment on a wide range of edge devices for arrhythmia monitoring.

INDEX TERMS Electrocardiogram, machine learning, edge inference, convolutional neural network, interpretable neural network, finite impulse response, short-time Fourier transform.

I. INTRODUCTION

Cardiovascular arrhythmias are a set of disorders characterized by irregular cardiac electrical activity. Arrhythmias such as ventricular fibrillation and flutter can cause cardiac arrest, hemodynamic collapse, and sudden death. Cardiovascular diseases induced by long-term cardiac arrhythmias are the leading cause of death globally, according to the WHO [1]. The intricacy of arrhythmias and their mechanical and clinical interrelationships causes numerous misdiagnoses and cross classifications using visual criteria. Moreover, clinical examination and diagnosis utilizing ECG data by physicians are time-consuming, impractical, and sometimes unavailable to remote places. Automatic arrhythmia beat categorization is thus urgently required for dynamic ECG processing.

Electrocardiography is still the most accessible and extensively used method for measuring cardiac electrical activity due to its simplicity, non-invasiveness, and low cost. The electrocardiogram (ECG) represents the electrical activity of the heart and provides vital information about heart function.

The associate editor coordinating the review of this manuscript and approving it for publication was Rajeswari Sundararajan.

Automatic ECG analysis is critical in cardiac monitoring, especially long-term monitoring with huge amounts of data. Arrhythmias are often brief in duration and cannot be discovered by physical examination or standard ECG due to time constraints. Longer ECG recording periods are required to detect arrhythmias, analyze their link to patient symptoms, or test the success of medications.

Many approaches have been proposed for collecting long ECG records [2]. For instance, a Holter monitor is used to collect ECG data using a traditional tape recorder or solid-state storage device, which is then processed and displayed for physician examination. The patient is instructed to keep a symptom diary and to record the time on the Holter clock when symptoms occur. Another option is to monitor patients via telemetry in the hospital, but this has serious drawbacks, including low patient acceptance. Mobile cardiac outpatient telemetry (MCOT) systems that allow for multiple days of ECG monitoring have been created. On the other hand, episodic monitors can capture ECGs during symptoms allowing patients to record ECGs, save them, and then fax them to doctors. Automatically-activated monitors which start recording when detecting an irregular heart rhythm have been advanced to replace the manually-activated monitors in which patients had to activate the device fast to acquire ECG recordings while experiencing symptoms Finally, many individuals with recurrent loss of consciousness or severe symptoms can have a device implanted beneath the skin that records information for later recovery. Modern pacemakers and implantable defibrillators can also collect information regarding arrhythmias for later retrieval. The pacemaker's signals can be recorded and analyzed later to confirm or diagnose an arrhythmia.

Unfortunately, all the aforementioned approaches for ECG monitoring neither provide an instantaneous diagnosis of the ECG root cause nor suggest immediate medical intervention. The alternative is continuous monitoring of the ECG activity in real-time and deploying automatic ECG classifiers for early detection and identification of sudden heart arrhythmias. Nowadays, such an approach can be efficiently deployed depending on recent advancements in the automatic Artificial Intelligence (AI) ECG classification methods, cloud services, and wearable technology. A single-lead ECG chest belt can be used to measure the ECG signal and send it wirelessly via the internet to a cloud service running an ECG arrhythmia detection model for long-term ECG rhythm monitoring and arrhythmia detection. The main concerns with this approach are the privacy of the patient, latency of the internet connection, connectivity of the ECG sensor, and availability of the cloud service.

In this work, we propose ECG classification and arrhythmia detection at the edge to address the previous concerns. Instead of relying on a cloud service for arrhythmia detection, a microcontroller-based edge device is used for acquiring the ECG signal, detecting heart arrhythmia in real-time, and alarming the patient to immediately take measures. However, edge deployment of AI models is a challenging task due to the demanding requirements of modern AI algorithms and the computation constraints of edge devices. Moreover, the criticalness of arrhythmia detection for the patient's life necessitates increasing the automatic detection accuracy which introduces an extra challenge.

To address the above challenges, some guidelines have been applied to the proposed classifier. The internationally-accepted MIT-BIH arrhythmia database is used for training and testing the ECG classifier. A single lead will be employed to capture the ECG signal to facilitate its usage by the patient. Due to its recent advancements, a deep neural network (DNN) model is used for ECG classification. The time-domain sampled ECG signal will be fed directly to the DNN model without further preprocessing or feature engineering. The real-time performance of the proposed model was planned in advance to fit the resource constraints of edge inference. The DNN model is optimized for edge deployment by applying state-of-the-art weight quantization and pruning techniques. Finally, the model is extensively tested on the edge device to verify its functional correctness.

A convolutional neural network (CNN) composed of a cascaded stack of 1D and 2D convolutional (Conv1D and Conv2D) layers and dense layers is developed. We provide a clear interpretation of the 1D convolutional layer (Conv1D) as a Finite Impulse Response (FIR) and exploit this interpretation to develop a short-time Fourier Transform (STFT) layer to extract the spectrogram of the input ECG signal. To the best of our knowledge, this is the first work to provide a clear interpretation of the Conv1D layer as a frequency-selective FIR filter. The Conv1D layer kernels are designed as a bank of adjacent FIR band-pass filters (BPFs) acting as an STFT computation engine. The Conv1D feature maps produced by the FIR filter bank are then reshaped into a 2D heatmap image to be fed to a Conv2D CNN classifier. The advantage of such an approach compared to using a pre-processing STFT computation stage commonly used in the literature is that our approach produces a lightweight self-contained CNN model amenable to edge optimization.

Four model variants are developed, tuned, trained, and tested on a cloud server. The testing results show that the proposed models achieve comparable classification results including accuracy, recall, precision, and F1-scores compared to the state-of-the-art ECG classifiers. The developed models are then optimized using post-quantization and training-aware quantization methods for edge deployment. Finally, the optimized models are tested and benchmarked on a raspberry-pi device. The proposed models achieve significant classification results using minimum computation resources fitting the computational constraints of the edge device.

The main contributions of this work include:

- Advancing a novel CNN topology for time series data tailored and optimized for edge inference.
- Providing clear interpretation of the Conv1D layer as a finite impulse response (FIR) frequency-selective filter and visualizing the Conv1D layer feature maps.

- Testing the ECG classifier on an edge device and reporting its performance and benchmarking results and comparing our work to recent state-of-the-art ECG classification methods and showing its competence.

The remaining of this paper is organized as follows: In Section II, a brief background of the automatic heart monitoring and classification literature and their related work are presented. A brief introduction to the MIT-BIH arrhythmia database and how it is employed in this work is presented in Section III. CNNs and their interpretation as FIR filters are discussed in Section IV. Methods and tools used in this work are advanced in section V. Model testing results on the cloud and edge machines and a comparison between the proposed model and state-of-the-art ECG classification models in addition to visualization of the Conv1D activation and feature maps are presented in Section VI. Conclusions and future work are portrayed in Section VII.

II. LITERATURE REVIEW

A typical ECG waveform consists of a P wave, QRS complex wave, and T wave as shown in Figure 1, which reflect electrical activities of depolarization and repolarization processes of the atria and ventricle [3]. Each heartbeat contains a series of deflections away from the baseline on the ECG that reflect the time evolution of the heart's electrical activity. P-wave is a small defection caused by atrial depolarization; Q, R, and S waves are known as the QRS-complex, which is the largest-amplitude portion of the ECG, caused by ventral depolarization; T-wave is caused by ventral polarization. Up to 12 separate leads can be used to measure ECG including three bipolar limb leads, three unipolar limb leads, and six unipolar chest leads. Figure 1 depicts ECG measured by the modified limb lead II (MLII) and chest lead V1. Each lead illustrates the electrical activity of the heart from a particular angle across the body. The normal heart rhythm is called sinus rhythm in which the triggering impulses propagate throughout the four chambers of the heart in a coordinated manner. The abnormal heart rhythms are called arrhythmias, which occur due to changes in the normal sequence of electrical impulses of the heart. The ECG can be used to spot and identify several types of arrhythmias. Figure 1 depicts the ECG of a normal sinus rhythm and ventricular arrhythmia.

The main stages of automatic heartbeat monitoring using the ECG are: data acquisition, preprocessing, feature engineering, and ECG signal classification [4]. The MLII is the most commonly used lead for ECG data acquisition and arrhythmia detection as it highlights various segments within the heartbeat including the P, QRS, and T waves [3]. The ECG preprocessing stage includes filtering unwanted signal components such as baseline wandering and power line interference, signal denoising, segmentation, and QRS complex detection. Myriad ECG processing techniques have been proposed in the literature with a wide range of complexity and performance [4], [5].

Feature engineering is the stage in which important features of the ECG are extracted and prepared for the classification stage. A feature is any information collected from data that can be used to categorize it. The features can be retrieved directly from the ECG signal morphology or from the heart rhythm in various forms. The most common feature used in the literature is the RR interval which is correlated with the variations in the ECG curve morphology and can be calculated from the cardiac rhythm. The RR interval is the time between the R peak of one heartbeat and the R peak of another heartbeat. Alterations in the RR interval are linked with changes in the ECG waveform shape caused by arrhythmias. Other morphological features such as intervals between the fiducial points of the heartbeat including P wave duration; QRS complex interval; and PP, ST, TP segment intervals are also used.

Time domain, frequency domain, time-frequency domain, and statistical techniques are commonly used to capture the significant features of the heartbeats [4]. Statistical approaches are often employed to extract relevant features from heartbeats in the temporal domain [6]. The classical Fourier transform (FT) is also used to obtain the ECG frequency spectral features however it can only capture global frequency information decoupled from their occurrence time. The ambiguity of FT is overcome by STFT in which the FT is repeatedly computed for a fixed-length moving temporal window to provide local time-frequency information or spectrogram of the signal however a trade-off arises between time-frequency resolution. The shortcoming of STFT is overcome by Wavelet transform (WT) in which dilated versions of a mother wavelet are shifted and correlated with the ECG signal to extract a high-resolution time-frequency 2D image called scalogram of the signal. Both continuous WT (CWT) and discrete WT (DWT) have been extensively used for ECG preprocessing and feature extraction [7]. The approaches that provided the highest accuracy in the literature used features from the time/frequency domain and the RR interval.

For a 1D input ECG signal, the sample points of the heartbeat signal can be used directly as features in 1D CNNs which are known for their capability of automatic feature extraction. The time-frequency spectrograms and scalograms of ECG segments obtained using STFT and CWT, respectively, can also be used as input images to 2D CNNs for feature extraction and classification. 2D CNNs are more prevalent with well-established models due to their wide usage for image applications. The advantage of this approach is eliminating the need for cardiology experts and relying on the automatic power of CNN for extracting ECG features that maximize the classification accuracy.

The final and most important stage in ECG monitoring is the classification stage. Generally, machine and deep learning methods have been extensively investigated for this task [8], [9]. The most commonly used methods are support vector machines (SVMs) and deep neural networks (DNNs). SVMs models with various feature types have been extensively used for ECG classification [3], [5], [6], [10] yet such models suffer from the computational complexity of the SVM algorithm. On the other hand, many recent works proposed

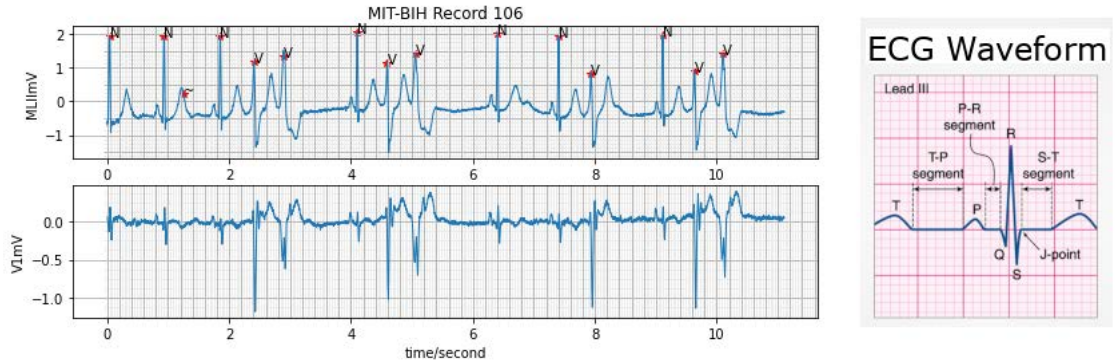


FIGURE 1. Sample of an ECG signal from the MIT-BIH database with a normal sinus rhythm and ventricular arrhythmia & The ECG wave segments.

various topologies of 1D and 2D CNNs in conjunction with different feature spaces for ECG classification [3], [5], [11], [11], [12], [13].

Alqudah *et al.* [14] presented a comparative study between different ECG time-frequency representations including Log-Scale STFT, Mel-Scale DWT, Bi-spectrum, and Third-order Cumulant and various CNN architectures including AOCT-NET, Mobile-Net, Squeeze-Net, and Shuffle-Net. Models are trained and tested using a subset of the MIT-BIH arrhythmia database representing six different heartbeat classes. The best overall performance among all used CNN architectures was MobileNet with an overall accuracy of 93.8%, while the best spectrum representation among all used was the bispectrum with an overall accuracy of 93.7%. It has been shown that the spectrum representations of ECG beats have significantly improved the classification results.

Cao *et al.* [15] proposed a transfer learning 2D CNN model for ECG classification using the STFT representation of the heartbeat. The MIT-BIH database is used for model training and testing and the ResNet-18 image classification model is fine-tuned to classify the ECG STFT spectrograms. Huang *et al.* [16] advanced a 2D CNN for ECG classification using the STFT spectrograms of the heartbeat signals. Models are trained and tested using a subset of the MIT-BIH arrhythmia database representing five different heartbeat classes. The classification results show that the proposed 2D-CNN model can reach an average accuracy of 99.00%. Ullah *et al.* [17] proposed a 8-class 2D CNN ECG classifier using the STFT spectrogram representation of the heartbeat signals. The proposed model consisting of four convolutional layers and four pooling layers is evaluated on the MIT-BIH arrhythmia dataset and it achieved 99.11% classification accuracy. Wu *et al.* [13] developed a CNN model to classify ECG heartbeats using various time-frequency distributions including STFT, CWT, and pseudo-Wigner-Ville distribution (PWVD). Models are trained and tested using a subset of the MIT-BIH arrhythmia database representing 12 rhythm classes. The method using STFT and a CNN achieved the best classification performance.

The common theme between these works is using the STFT spectrogram accompanied by 2D CNN models to enhance the ECG classification results. In all related STFT-based ECG classifiers, the STFT computation is carried out as a pre-processing step using a separate computational block which adds a computation overhead to the classifier model. The main feature of our proposed model is implementing the STFT extraction procedure as a part of the CNN model itself exploiting the FIR interpretation of the Conv1D layer. Such an approach results in a self-contained classifier model with reduced computational requirements that fits edge inference.

Many recent works other than the STFT-based CNNs have been also proposed for ECG classification and arrhythmia detection. Cui *et al.* [10] proposed an ECG feature extraction method based on DWT and Conv1D with Principle Component Analysis (PCA) to reduce the number of features. An SVM classifier is trained on an upsampled subset of the MIT-BIH dataset to address dataset imbalance.

Li *et al.* [18] presented an ECG classifier based on Incremental Broad Learning (IBL) and biased dropout. Incremental learning is a machine learning paradigm where the learning process takes place whenever new examples emerge and adjusts what has been learned according to the new examples. Baseline wander filtering and DWT are used for ECG signal denoising, and morphological and rhythm features such as RR intervals are extracted from the denoised ECG segments. An IBL DNN is used for ECG classification.

Liu *et al.* [19] proposed several models for ECG classification based on Wavelet Scattering Transform and PCA for feature extraction and a variety of NN and probabilistic NN (PNN) and k-nearest neighbor (KNN) classifiers for ECG classification. WST builds translation invariant, stable, and informative signal representations WST is implemented with a CNN with preassigned weights that iterates over traditional WT, nonlinear modulus, and averaging operators. In PNN, the class probability of a new input data is estimated and the Bayesian rule is then employed to allocate the class with

the highest posterior probability to new input data. KNN is a non-parametric supervised classification algorithm that classifies test data by measuring its Euclidean distance in the feature space to all labeled training samples and returns the nearest K labels and assigns the most frequent label to the test data.

Mousavi and Afghah [20] proposed an ECG classification method based on the CNN and Bidirectional Long-Shirt Term Memory (BiLSTM) sequence to sequence auto-encoder. A Conv1D CNN extracts a set of features from the given ECG heartbeats, then, a BiLSTM encoder maps the features to a sequence capturing data temporal patterns, and finally, the decoder takes the sequence representations and produces the classification probabilities The Synthetic Minority Oversampling Technique (SMOTE) upsampling algorithm is used to address the class imbalance. The model is tested for both intra- and inter-patient schemes and the reported scores are among the best scores reported for the ECG classification works. Generally, Recurrent-Neural Networks (RNNs) and LSTM, specifically, suffer from increased computational loads and computation time limiting their applicability to edge inference.

Raj and Ray [6] proposed a personalized arrhythmia monitoring platform for real-time detection of ECG arrhythmias. Discrete Orthogonal Stockwell Transform (DOST) is used for time-frequency feature extraction and the Artificial Bee Colony (ABC) optimized twin least-square support vector machine (LST-SVM) algorithm is used for signal classification. Two median filters and a 12-tap FIR LPF are used for baseline wandering removal and high-pass noise and power-line interference filtering. DOST is computed for a 1D time-series signal by applying Fast FT (FFT), multiplicative windowing, and inverse FFT (IFFT) to yield a set of coefficients representing the time-frequency morphological features of the signal. The ABC-LSTSVM is an SVM algorithm with reduced complexity to fit embedded device constraints.

Acharya *et al.* [21] presented a CNN for ECG classification. DWT is used for noise removal and a 9-layer Conv1D CNN is employed for signal classification. A custom upsampling method is proposed for addressing the class imbalance. The accuracy, precision, and recall results are acceptable but not the best compared to other models presented in the literature. Nevertheless, the proposed model is a good fit for edge deployment due to the minimum preprocessing steps applied except the DWT noise removal step.

The main purpose of this work is to develop a highly accurate ECG classification and arrhythmia detection model at the edge. The main challenge is meeting the accuracy requirements of the ECG classification problem using a resource-constrained edge microcontroller or embedded system device. A common feature of related ECG classifiers is using computationally intensive feature extraction stages and DNN models which are not optimized for edge computing. Related ECG classification works do not address minimizing the computation requirements of the developed models which is essential for edge deployment of AI models. None of the related work presented real-time performance analysis of the developed models or an interpretation of the underlying DNN model. Other parameters such as the model size, number of parameters, memory usage, and training time are neither optimized nor presented in the literature, as well. In this work, we aim to address the above challenges and provide a lightweight interpretable ML model for ECG classification and arrhythmia detection ready for edge deployment.

III. DATASET AND FEATURE SELECTION

The internationally recognized standard ECG databases include the MIT-BIH database, the AHA (American Heart Association) database, the European Community CSE database, and the European ST-T database. The MIT-BIH arrhythmia database is the most commonly utilized and highly acknowledged database in the academic world [3]. The MIT-BIH Arrhythmia Database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects collected between 1975 and 1979 [22], [23]. Twenty-three recordings were chosen at random from a set of 4000 24-hour ambulatory ECG recordings collected from a mixed population of inpatients (about 60%) and outpatients (about 40%); the remaining 25 recordings were selected from the same set to include less common but clinically significant arrhythmias that would not be well-represented in a small random sample. The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range. Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database. A total of 15 annotations are assigned to the R-peaks of the ECG beats.

According to the standard developed by the Association for the Advancement of Medical Instrumentation (AAMI) [24], 17 recommended arrhythmia categories are classified into 5 essential groups or superclasses. Following the AAMI-recommended practice, four-paced recordings are not used. The AAMI standard emphasizes the problem of distinguishing ventricular ectopic beats (VEBs) from non-ventricular ectopic beats, and hence normal and arrhythmic beats are remapped to the five AAMI heartbeat classes. We followed the AAMI standard which is commonly used in the literature [3] aiming to standardize the evaluation process considering a clinical point of view and AAMI recommendations and to ensure a fair comparison with the results in the related literature. Annotations in the MIT-BIH dataset are mapped to five different beat categories acting as dataset labels following the AAMI standard as shown in Table 1.

Each ECG record of the MIT–BIH Arrhythmia Database includes two leads originating from different electrodes. The most common leads in the database are MLII and V1. To maintain the consistency of leads, only MLII lead readings are used in this research. Records 102 and 104 have

TABLE 1. Mapping of MIT-BIH arrhythmia types and Advancement of Medical Instrumentation (AAMI) classes.

AAMI Classes	Normal Beat (N)	Supraventricular Ectopic Beat (S)	Ventricular Ectopic Beat (V)	Fusion Beat (F)	Unknown Beat (Q)
	Normal beat (NOR)—N	**Atrial premature beat (AP)—A**	**Ventricular escape beat (VE)—E**	**Fusion of ventricular and normal beat (fVN)—F**	**Unclassifiable beat (U)—Q**
	72970	2536	106	799	15
MIT-BIH Arrhythmia Types	Right bundle branch block beat (RBBB)—R	Premature or ectopic supraventricular beat (SP)—S	Premature ventricular contraction (PVC)—V		Fusion of paced and normal beat (fPN)—f
	7259	2	7081		260
Number of beats per class	Left bundle branch block beat (LBBB)—L	Nodal (junctional) premature beat (NP)—J			Paced beat (P)—/
	8075	81			3620
	Atrial escape beat (AE)—e	Aberrated atrial premature beat (aAP)—a			
	16	150			
	Nodal (junctional) escape beat (NE)—j				
	229				
Total Number of beats	88549	2769	7187	799	3895

been excluded from the dataset because they do not contain the MLII lead readings. The number of beats provided in Table 1 does not include records 102 and 104. Figure 1 shows a sample of an ECG signal from the MIT-BIH record 106 with normal and premature ventricular contraction beats.

To prepare the dataset for machine learning, the ECG signals are downsampled to 128 samples/sec to reduce the DNN computation load. The ECG signals from various records are segmented on a beat-by-beat basis. The heartbeats are segmented by filtering out non-beat annotations from the dataset and extracting 0.5-second segments (64 samples) centered at the annotated R-peak. Segmented beats with less than 0.5 s intervals are padded to make all heartbeats of the same length, which is required for the DNN model. No filtering or preprocessing stages have been applied that assume any knowledge about the signal shape or spectrum. It has been shown that the use of normalized RR-intervals significantly improves the classification results [25]. The preceding and subsequent RR peak intervals have been also extracted and normalized to the sampling frequency to be used as input features to the DNN models. The recommended beat extraction method is simple and can be directly applied at the edge device using a simple peak detector.

The MIT-BIH dataset signals were extracted from Holter recordings and filtered to limit analog-to-digital converter (ADC) saturation and for anti-aliasing, using a band-pass filter from 0.1 to 100 Hz [23]. In this work, we will be sufficing with this filter for noise removal during model training and testing, *i.e.* we will use the MIT-BIH heartbeat signals from the training and testing datasets without further pre-processing. For model deployment on the edge device, a pre-processing baseline wandering and noise removal stage will be implemented using a low-computational cost FIR filter [25] or moved to the ECG analog front-end to reduce the computation load at the edge device. Some related works use DWT for noise removal [3]. However, such approaches incur additional computation overhead to the classification model limiting their usage on resource-constrained edge devices.

The total number of segmented annotated beats is 103200 due to excluding records 102 and 104. Three protocols are proposed in the literature for dividing the MIT-BIH dataset into training and test sets: intra-patient, inter-patient, and random division schemes [3]. In the intra-patient division scheme, the heartbeats from the same patient are used for training and testing which makes the evaluation process biased. In the inter-patient division scheme proposed in [26], the training and test datasets are divided by the record numbers such that heartbeats within each set come from different individuals eliminating the evaluation process bias. In this work, the random division scheme is used to ensure keeping the dataset distribution statistics in both the training and testing sets while eliminating the evaluation bias. The MIT-BIH database is randomly shuffled, stratified, and split into training and test datasets with a splitting ratio of 25%. The training dataset is further split into training and validation datasets with a splitting ratio of 25%. The total numbers of annotated heartbeat examples in the training, validation, and test datasets are 58050, 19350, and 25800, respectively.

IV. CONVOLUTIONAL NEURAL NETWORKS (CNNs) AND FINITE IMPULSE RESPONSE (FIR) FILTERS

CNN is a deep learning model for processing data with a grid pattern, such as photographs, that is inspired by the architecture of the human visual cortex and designed to learn spatial hierarchies of characteristics automatically and adaptively, from low- to high-level patterns. CNN is a mathematical construct made up of three types of layers (or building blocks): convolutional, pooling, and fully connected layers. The first two layers, convolution, and pooling extract features while the third, a fully connected layer, uses the extracted

features for classification. Deep CNN features multiple cascaded stacks of Convolutional and Pooling layers to increase their expressiveness power. A 1D convolutional (Conv1D) layer slides several kernels across a time-series sequence or signal, producing a 1D feature map per kernel. The sliding shift amount is determined by the strides parameter which specifies the stride length of convolution. Each kernel will learn to detect a single very short sequential pattern (no longer than the kernel size). The inactivated Conv1D layer output is expressed as follows [27]:

$$y_k^l = b_k^l + w_k^{l-1} \circledast x^{l-1} = b_k^l + \sum_{i=1}^{N_{l-1}} w_{ik}^{l-1}\, x_i^{l-1} \qquad (1)$$

where y_k^l is the layer output, b_l^k is the bias of the k^{th} neuron at layer l, x_i^{l-1} is the output of the i^{th} neuron at layer $l-1$, w_{ik}^{l-1} is the kernel weight from the i^{th} neuron at layer $l-1$ to the k^{th} neuron at layer l, N_{l-1} is the size of the Conv1D kernel at layer $l-1$, $\circledast$ is the 1D correlation operator [27].

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response is of finite duration [28]. For a causal FIR filter of order N, each value of the output sequence is a weighted sum of the most recent input values as shown in Figure 2. The FIR output is defined as:

$$y[n] = h[n] * x[n] = \sum_{i=0}^{N} x[i]\, h[n-i] \qquad (2)$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, N is the filter order; an N^{th}-order filter has $N+1$ terms on the right-hand side, h_i is the value of the impulse response at the i^{th} instant for $0 \le i \le N$ of an N^{th}-order FIR filter, and $*$ is the 1D convolution operator [27]. If the filter is a direct form FIR filter then $h[n]$ is the filter coefficients or taps. This computation is also known as discrete convolution.

Correlation is equivalent to convolution with the time-reversed impulse response of the Conv1D kernel where $h[n] = w[-n]$ [29]. Comparing the discrete convolution computed by the FIR filter and the Conv1D kernel operation illustrates that the discrete convolution is equivalent to the inactivated Conv1D kernel convolution between the layer weights and the time-reversed version of the kernel weights for the bias term $b = 0$. The time delay or shifting operation of the FIR filter is equivalent to sliding the Conv1D kernel across the input signal. A Conv1D kernel with a single stride computes the discrete convolution between the 1D input signal and the Conv1D kernel in parallel. In other words, the Conv1D kernel acts as an FIR filter applied to the 1D input signal. Accordingly, the Conv1D filter kernel is equivalent to the impulse response of the FIR filter where the kernel weights are the filter coefficients or taps of the direct form realization of the FIR filter. Note that a Conv1D kernel of N_k length has an order of $N = N_k - 1$ as an FIR filter.

The convolution theorem states that under suitable conditions the Fourier transform of a convolution of two functions

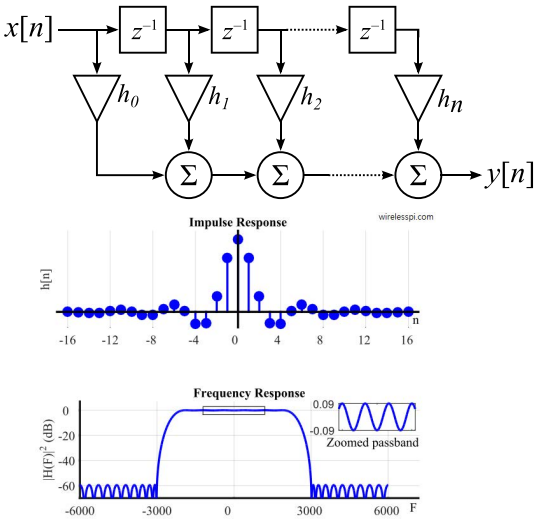


FIGURE 2. Direct form discrete-time FIR filter of order N. The top part of the block diagram is an N-stage delay line with $N+1$ taps. Each unit delay is a z^{-1} operator in $\mathcal{Z}$-transform notation & An example of the FIR filter impulse response $h[n]$ and power spectrum $|H(f)|^2$ for $N = 32$.

(or signals) is the point-wise product of their Fourier transforms [28]. The filter output for an input sequence $x[n]$ is described in the frequency domain by the convolution theorem as follows:

$$\underbrace{\mathcal{F}(x * h)}_{Y(\omega)} = \underbrace{\mathcal{F}(x)}_{X(\omega)} \cdot \underbrace{\mathcal{F}(h)}_{H(\omega)},$$
$$\text{and} \quad y[n] = x[n] * h[n] = \mathcal{F}^{-1}\{X(\omega) \cdot H(\omega)\} \qquad (3)$$

where operators $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the discrete-time Fourier transform (DTFT) and its inverse, respectively. The complex-valued, multiplicative function $H(\omega)$ is the filter's frequency response. Figure 2 illustrates an example of the filter impulse response $h[n]$ and power spectrum $|H(f)|^2$ of an FIR Low-pass Filter (LPF) for $N = 32$.

Consequently, the Conv1D kernel operation is equivalent to applying an FIR frequency-selective filter to the input signal. Furthermore, since the Conv1D kernel works as a sliding window along the time axis, the filter output is a function of time as well. Therefore, the Conv1D kernel output, which is also known as the feature map, is a time-domain signal that indicates the existence of specific frequency components in the signal at specific time instants. The Conv1D kernel weights can be pre-designed and assigned as non-trainable parameters to apply specific FIR filtering operations but they also can be trained to learn significant features of the signals as usually done in CNNs. Nonetheless, a trainable Conv1D kernel is still interpreted as an FIR filter with learned parameters since the kernel operation is invariant. The Conv1D layer comprises multiple kernel filters which output N_f feature maps each of N_s length where N_f is the number of Conv1D filters and N_s is the number of samples of the input signal. The Conv1D feature maps collectively can be grouped and treated as a 2D heatmap that exhibits the time-frequency features of

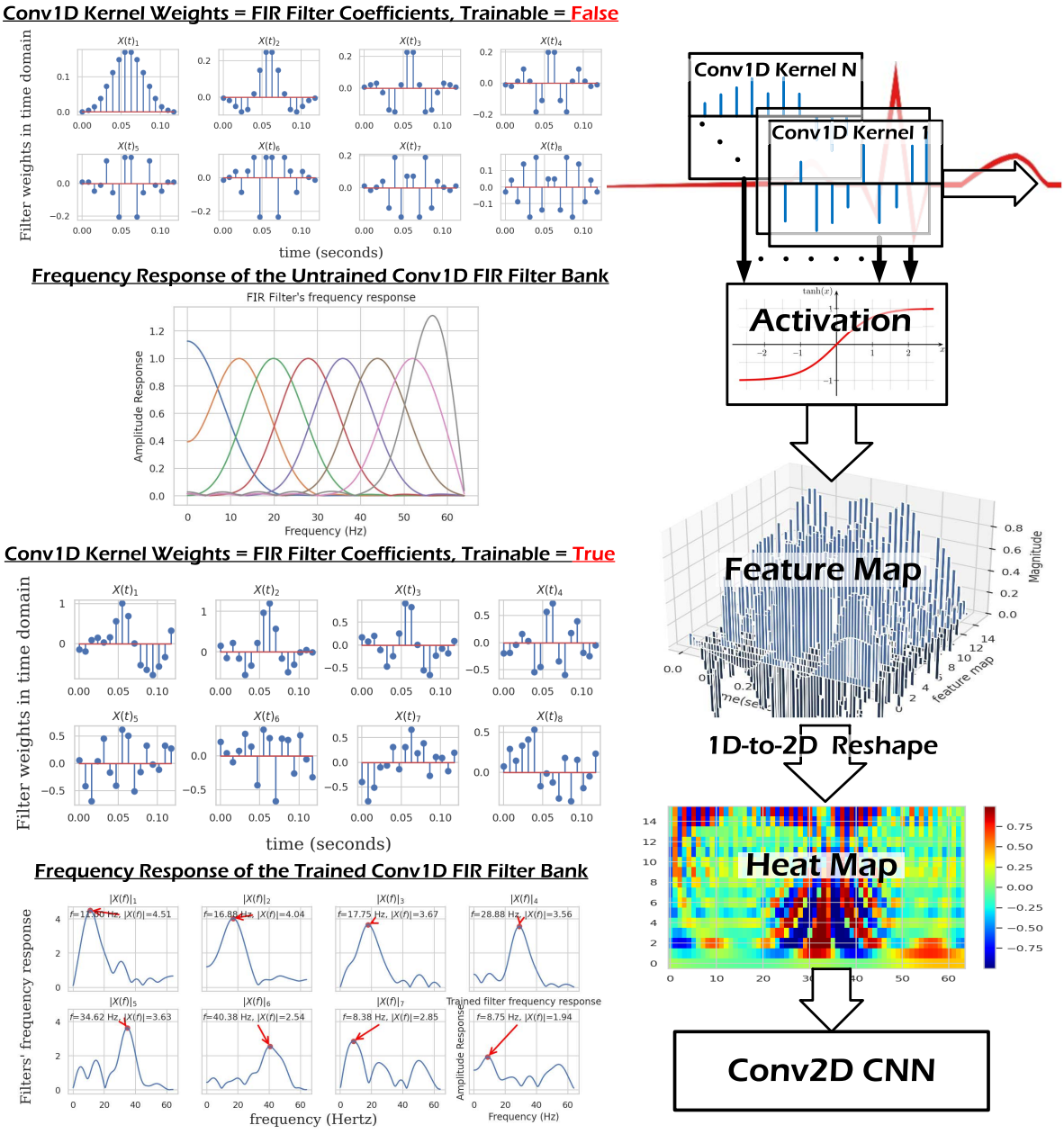


FIGURE 3. Self-contained STFT CNN architecture and the impulse and frequency responses of the FIR filter bank.

the signal. The produced 2D heatmap can be classified using a Conv2D CNN as shown in Figure 3.

The Conv1D kernels can be designed to perform any kind of convolution-based functions or FIR filtering techniques including frequency-selective and matched filters by controlling the kernel length N_k and assigning appropriate filter coefficients to the kernel weights. Figure 3 shows the Conv1D layer kernel weights and frequency response of a bank of FIR BPFs ($N_F = 8$) with equal bandwidths and equally-spaced adjacent center frequencies. The 2D heatmap image resolution is $N_s \times N_f$. The proposed Conv1D layer can be designed to imitate the operation of STFT, CWT, or other time-frequency transforms by assigning pre-designed filter weights to the Conv1D FIR kernels. STFT computes the signal spectrogram by repeatedly evaluating FT of the signal over a short period of time and displaying the spectrum versus time. The CWT applies a set of filters derived from dilated versions of the mother wavelet to the signal to compute the signal scalogram. Moreover, the proposed Conv1D layer can be designed as an LPF, HPF, or BPF for ECG noise removal without the need for extra preprocessing stages.

The motivation for inventing this approach is to enhance the performance of the ECG CNN classifier by designing a self-contained STFT-based 2D CNN without applying preprocessing computationally-intensive algorithms such as STFT or WT to meet the edge device computational constraints. This approach can be applied for multi-lead ECG classification as well by dealing with the signal of each lead as a separate input channel and using multi-channel Conv2D CNN for classification. Another reason for devising such an approach is that existing DNN design tools do not support the quantization of custom preprocessing functions such as STFT or WT for edge computing. The proposed FIR-based STFT Conv1D layer can be readily quantized using the existing tools. Furthermore, the proposed approach provides a clear interpretation of the Conv1D CNN operation as a frequency-selective filter bank which is a novel contribution.

V. METHODS AND TOOLS

The MIT-BIH dataset is highly imbalanced as shown in Table 1 with a majority class to minority class ratio of 25.8 and the ratio of the normal ECG beats to the total number of beats is 86%. Addressing class imbalance with traditional machine learning techniques has been studied extensively over the last two decades. Methods for handling class imbalance are grouped into data-level techniques, algorithm-level methods, and hybrid approaches [30]. Data-level methods for addressing class imbalance include over-sampling and under-sampling while algorithm-level methods is handling class imbalance by adjusting the learning or decision process in a way that increases the importance of the minority class. In this work, the class imbalance problem is addressed at the algorithm level by incorporating the class-weight parameter in the model training process to assign higher weights for minority classes during loss function optimization which is equivalent to oversampling the minority class. The Adam optimizer with adaptive learning rate scheduling initiated at 0.01 is used for model training.

Keras with the Tensorflow backend is used to train and test the CNN classifiers. Keras is an open-source software library that provides a Python interface for the TensorFlow library. TensorFlow is an open-source framework for machine learning created by Google with a comprehensive, flexible ecosystem of tools, libraries, and community resources that help developers easily build and deploy ML-powered applications. The development flow and tools used in this work are presented next.

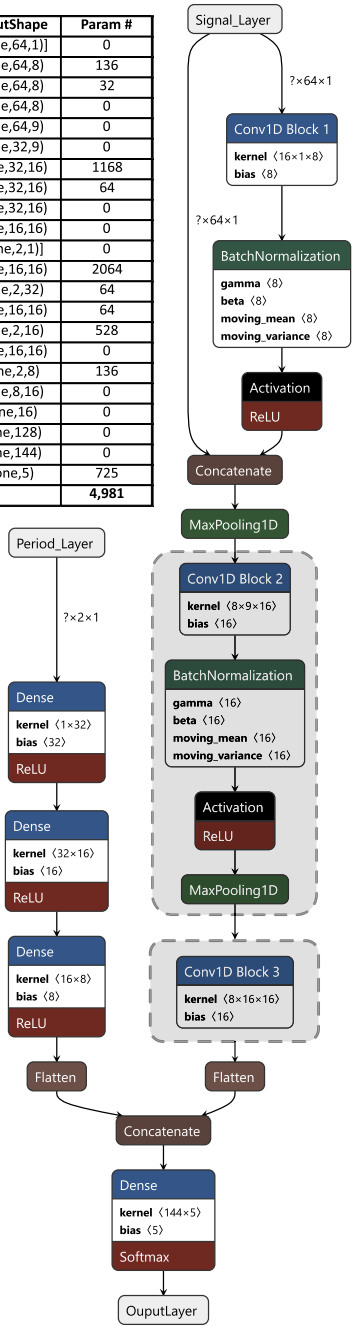
1) Keras Tuner (a framework for optimizing hyperparameter search) is used to optimize the model hyperparameter search process. Hyperparameters include all non-trainable parameters of the model and their tuning is very challenging and time-consuming. In this work, the hyperparameter search space includes the number of convolutional layers, kernel size of each layer, number of filters in each layer, activation function selection from Relu, Tanh, and Sigmoid, and Boolean parameters to include or not regularization layers such as Dropout and BatchNormalization layers. Keras Tuner comes with the Bayesian Optimization, Hyperband, and Random Search algorithms. The three algorithms have been investigated and the Hyperband algorithm is found to give better results for the given dataset. Bayesian-optimized models tend to overfit the training set resulting in a significant variance. The hyperband algorithm is a combination of random search with adaptive resource allocation and early stopping that accelerate the hyperparameter search process.

Hyperparameter search is approached as an optimization problem with the objective of minimization/maximization of a specific quantity. Usually, maximizing the validation accuracy is the main objective of classification algorithms. However, due to the class imbalance nature of the training dataset, maximizing the model accuracy does not tend to give the best results in terms of detecting irregular heart activities due to the dominance of the normal class in the dataset. For example, if a classifier is set to predict all beats as normal it would achieve 86% accuracy with all normal beats being correctly classified and all other beats being misclassified. The validation Area Under the Curve of the Receiver-Operating Characteristics (ROC-AUC), the validation recall score, and the validation F1-score have been inspected as the optimization objectives. In our experiments, the F1-score with macro averaging tends to give the best results in terms of maximizing the classification accuracy of the minority classes. Unfortunately, the hyperparameters found by Keras Tuner cannot be used directly to develop the edge models because Keras Tuner does not consider optimizing the model complexity while searching for the best parameters. The Keras Tuner parameters are used as a guideline while developing the classifier models to be exported to the edge device.

2) Manual tuning of the proposed models is conducted to maximize the model F1-score while minimizing the model complexity. In this step, the model hyper parameters including the number of layers, the used regularization layers, the loss function, the loss optimizer, the dataset class imbalance mitigation method, and the search objective are drawn from the Keras Tuner step. The number of filters and kernel size of Conv1D and Conv2D layers is manually tuned to apply the filtering operations described in Section IV and reduce the model complexity. We designed N_F adjacent FIR Filter bank of BPFs each of order $N = N_k - 1$ with equal bandwidths and equally-spaced center frequencies between 0 and 64 Hz using the Hamming window method [28]. Figure 3 shows the frequency response of the FIR filters for $N_F = 8$ and $N_k = 16$. A Conv1D input layer is instantiated with N_F filters each of N_K size in which the FIR filter coefficients are assigned to the layer kernel weights.

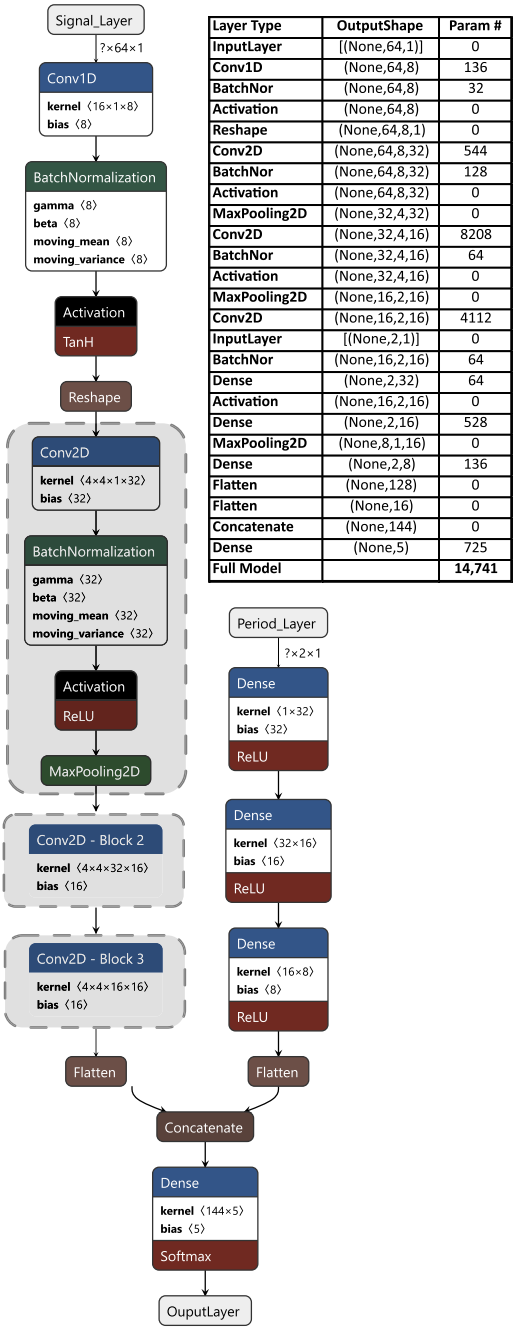
Two CNN models have been developed: a fully Conv1D model and a mixed Conv1D-Conv2D model in which the input layer of both models is the FIR Conv1D layer as shown in Figure 4. The fully Conv1D model is mainly developed to compare its performance to the proposed Conv1D-Conv2D model. For each model two approaches are adopted for training the models by switching the Boolean trainable parameter of the Conv1D layer from False to True and initializing the

Layer type	OutputShape	Param #
InputLayer	[(None,64,1)]	0
Conv1D	(None,64,8)	136
BatchNorma	(None,64,8)	32
Activation	(None,64,8)	0
Concatenate	(None,64,9)	0
MaxPooling1D	(None,32,9)	0
Conv1D	(None,32,16)	1168
BatchNor	(None,32,16)	64
Activation	(None,32,16)	0
MaxPooling1D	(None,16,16)	0
InputLayer	[(None,2,1)]	0
Conv1D	(None,16,16)	2064
Dense	(None,2,32)	64
BatchNor	(None,16,16)	64
Dense	(None,2,16)	528
Activation	(None,16,16)	0
Dense	(None,2,8)	136
MaxPooling1D	(None,8,16)	0
Flatten	(None,16)	0
Flatten	(None,128)	0
Concatenate	(None,144)	0
Dense	(None,5)	725
Full Model		4,981



(a) Conv1D_T model architecture

Layer Type	OutputShape	Param #
InputLayer	[(None,64,1)]	0
Conv1D	(None,64,8)	136
BatchNor	(None,64,8)	32
Activation	(None,64,8)	0
Reshape	(None,64,8,1)	0
Conv2D	(None,64,8,32)	544
BatchNor	(None,64,8,32)	128
Activation	(None,64,8,32)	0
MaxPooling2D	(None,32,4,32)	0
Conv2D	(None,32,4,16)	8208
BatchNor	(None,32,4,16)	64
Activation	(None,32,4,16)	0
MaxPooling2D	(None,16,2,16)	0
Conv2D	(None,16,2,16)	4112
InputLayer	[(None,2,1)]	0
BatchNor	(None,16,2,16)	64
Dense	(None,2,32)	64
Activation	(None,16,2,16)	0
Dense	(None,2,16)	528
MaxPooling2D	(None,8,1,16)	0
Dense	(None,2,8)	136
Flatten	(None,128)	0
Flatten	(None,16)	0
Concatenate	(None,144)	0
Dense	(None,5)	725
Full Model		14,741



(b) Conv1D_Conv2D_T model architecture

FIGURE 4. **CNN model architectures.**

Conv1D layer weights with the FIR filter coefficients in both approaches. The proposed models are trained and tested on the MIT-BIH dataset and the average and per-class model metrics including accuracy, recall score, ROC-AUC score, and F1-score are measured and reported.

3) Models with the best F1-scores are then optimized for edge computation using the TensorFlow and TensorFlow lite (Tflite) optimization tools [31], [32] and the Google Qkeras package [33]. Quantization refers to techniques for performing computations and storing tensors at lower bit widths than floating-point precision. A quantized model executes some or all of the operations on tensors with integers or smaller float precision rather than the 32-bit single-precision floating-point. Quantization allows for a more compact model representation, smaller memory footprint, faster inference, and less-demanding computation requirements yet it comes at the expense of accuracy loss. Both quantization-aware training (QAT) [34], [35] and post-training quantization (PTQ) [36] techniques have been investigated to develop Tflite models ready for edge deployment. While the former

results in less reduction of the model accuracy, the latter does not need model training which can be much faster and easier to use. In QAT, weights and activations are fake quantized during both the forward and backward passes of training while in PTQ weights and activations of an already-trained model are quantized to a lower precision. QAT can be performed using either the TensorFlow optimization toolkit or the Qkeras library which provides more versatile quantization options including the quantizable layers and the quantization precision.

The quantized models are then subjected to weight pruning to remove superfluous values from weight tensors. Weight pruning reduces the number of parameters and operations in a calculation by deleting connections, and hence parameters, between DNN layers. The DNN parameters are set to zero to eliminate superfluous connections between neural network layers. Weight pruning is done during the training phase to allow the DNN to adapt to changes. The weight pruning API is developed on top of Keras simplifying its use with Keras models. Weight pruning can be performed simultaneously with QAT using either the TensorFlow optimization toolkit or the Qkeras library.

The optimized models are then converted to Tflite models for deployment on the edge device. Tflite is a package of tools that enables on-device inference of machine learning models. This package is composed of a runtime engine for ML model inference computation on edge devices and a set of tools for transforming and quantizing TensorFlow models post-training for usage on mobile and embedded devices. Tflite offers several PTQ options to choose from that fit the requirement of various computation platforms. In dynamic range and float16 quantization, weights are statically quantized from 32-bit floating-point to 8-bit integers (int8) and 16-bit floats (Float16), respectively. In full-integer quantization, both weights and activations of the model are statically quantized to int8. Edge inference using Tflite addresses five main concerns: latency (no round-trip to a server), privacy (no personal data is sent out of the device), connectivity (no internet connectivity is required), size (reduced model and binary size), and battery consumption (efficient inference and a lack of network connections).

4) The optimized Tflite models are finally exported to the edge device for testing and benchmarking. A raspberry-pi 3 model B+ with Cortex-ARMv8 64-bit SoC and 1GB DDR2 SDRAM running the Ubuntu 18.04 OS and hosting a Python 3.6 interpreter and Tflite runtime engine is used for this purpose. The ARM Cortex processor architecture inherently supports 32-bit integer and floating-point operations. The model metrics including accuracy, recall score, ROC-AUC score, and F1-score are measured for all Tflite models as well as the model performance metrics including the model size, memory usage, average inference time, and throughput.

VI. RESULTS AND DISCUSSION

Two CNN models are proposed: The first model comprises a stack of (Conv1D, BatchNormalization, Relu Activation, and MaxPooling) layers with the ECG signal fed to the input layer and another stack of dense layers fed with the normalized post- and pre- RR intervals. Outputs from both stacks are then flattened, concatenated, and fed to a dense layer with softmax activation to output the five ECG class probabilities as shown by Figure 4(a). The second model comprises a Conv1D input layer with Tanh activation for time-frequency feature extraction followed by a stack of (Conv2D, BatchNormalization, Relu Activation, and MaxPooling) layers with the ECG signal fed to the input layer and another stack of dense layers fed with the post- and pre-RR intervals. Outputs from both stacks are then fed to a dense layer with softmax activation to output the five ECG class probabilities as shown by Figure 4(b).

The best parameters of the Conv1D FIR input layer are found to be $N_F = 8$ and $N_K = 16$. This layer is fixed in both Conv1D and Conv1D_2D models and the parameters of the remaining layers are manually tuned to maximize the F1-score and minimize the number of model parameters. Two variants of each model are trained in which the Conv1D Trainable parameter is switched from False in the Conv1D_2D_T_FIR and Conv1D_T_FIR models to True in the Conv1D_2D_T and Conv1D_T models. The training process was conducted on a cloud machine featuring 8 CPU cores, 30 GB of RAM, and an NVIDIA QUADRO RTX 5000 GPU and hosted by the Paperspace Gradient cloud platform [37]. Experiments are repeated 10 times for each model and the average results are reported.

A. MODEL TESTING ON THE CLOUD

Table 2 shows the training and testing results of the proposed models on the cloud machine. The model number of parameters, size, training time, and GPU memory usage during training are illustrated. The training and testing accuracy of the developed models is depicted. The ROC-AUC, recall, precision, and F1- weighted and macro average scores are presented for the test set only. In macro average scores, class weights are not considered for calculating the average from individual class scores, unlike the weighted average which gives higher scores due to considering class weights. The model is tested using the cloud machine CPU and GPU and the average inference time and throughput are calculated. Throughput is calculated by dividing the number of test examples by the whole test dataset inference time.

The number of parameters is the same for both trainable and non-trainable Conv1D models yet the average inference time is greater in models with non-trainable parameters which can be attributed to that training the Conv1D layer results in sparse weight tensors which accelerates inference time. Comparing the training and test accuracy shows that the variance of all models does not exceed 1% indicating that the models do not overfit the training dataset and well generalize to the test dataset. Models with the trainable parameter of the Conv1D layer set to True outperform their counterparts which indicates that the initial FIR kernel weights have been updated during the backpropagation path of model training

TABLE 2. Classification and performance results of the proposed models for the training and test datasets on the cloud.

	Conv1D_2D_T_FIR	Conv1D_2D_T	Conv1D_T_FIR	Conv1D_T
Number of parameters	14741	14741	4981	4981
Training time (seconds)	190.92	178.63	218.64	262.64
Training GPU memory usage (MB)	223.75	308.5	100.75	100.5
Training Accuracy	99.66%	99.71%	99.41%	99.43%
Test Accuracy	98.77%	99.08%	98.65%	98.78%
ROC-AUC score	99.54%	99.74%	99.31%	99.52%
Average Recall Score – Weighted (Macro)	98.77% (91.34%)	99.08% (94.92%)	98.65% (93.02%)	98.78% (94.59%)
Average Precision Score – Weighted (Macro)	98.75 % (93.33%)	99.09% (93.78%)	98.68% (91.38%)	98.82% (91.93%)
Average F1-score weighted – Weighted (Macro)	98.76% (92.31%)	99.09% (94.34%)	98.66% (92.18%)	98.79% (93.21%)
CPU Inference time (msec)	0.25	0.16	0.09	0.08
CPU Throughput (inference/second)	4081.61	6214.42	11228.73	12954.85
GPU Inference time (μsec)	94.80	62.10	86.29	68.55
GPU Throughput (inference/second)	10548.90	16104.25	11588.40	14588.05
Model Size (KB)	292.38	298.51	160.10	165.45

TABLE 3. Per-class precision, recall, and F1-score metrics (%) of the proposed models for the test dataset only.

Model	**N**			**S**			**V**			**F**			**Q**			**Average**		
	PREC	**SEN**	**F1**	**PREC**	**SEN**	**F1**	**PREC**	**SEN**	**F1**	**PREC**	**SEN**	**F1**	**PREC**	**SEN**	**F1**	**PREC**	**SEN**	**F1**
Conv1D_2D_T_FIR	99.26	99.52	99.39	89.76	86.13	87.91	97.59	96.88	97.24	80.65	75.00	77.72	99.38	99.18	99.28	98.75	98.77	98.76
Conv1D_2D_T	99.62	99.52	99.57	90.52	92.49	91.49	97.88	97.50	97.69	81.82	85.50	83.62	99.08	99.59	99.33	99.09	99.08	99.09
Conv1D_T_FIR	99.47	99.20	99.34	84.80	90.32	87.47	96.56	96.88	96.72	77.18	79.50	78.33	98.87	99.18	99.03	98.68	98.65	98.66
Conv1D_T	99.59	99.23	99.41	85.35	92.63	88.84	96.66	96.72	96.69	79.07	85.00	81.93	98.98	99.38	99.18	98.82	98.78	98.79
Support		22137			692			1797			200			974			25800	

to achieve better classification results. The Conv1D_2D_T model achieves the highest test F1-score of 94.34% (the F1-score of the average cardiologist is 78% [38]) and accuracy of 99.08%. The Conv1D_T_FIR model has the smallest inference time and model size. Although the Conv1D_2D_T model accuracy is greater than the Conv1D_T model accuracy by only 0.3%, its F1-score is greater by 1.23% which illustrates the advantage of using Conv2D CNN for classification problems. In our experimentation, a trade-off is made between the model classification performance and the model complexity and real-time inference performance. Some models achieved better accuracy scores but were excluded due to the significant increase in the model size which would disallow their usage on edge devices.

Not only the average scores are used to select the best models but also the model complexity and the per-class detailed metric results have been employed. Figure 5 shows the normalized confusion matrix of the Conv1D_2D_T model.

In classification problems, model accuracy is defined as the percentage of true predictions to the total number of dataset examples. In terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), accuracy is defined as $\text{ACC} = (TP + TN)/(TP + TN + FP + FN)$. Class metrics including precision, recall, and F1-score are also measured. Precision is defined as the percentage of TP to the sum of TP and FP, $\text{PREC} = TP/(TP + FP)$, while recall or sensitivity is defined as the percentage of TP to the sum of TP and FN, $\text{SEN} = TP/(TP + FN)$. For arrhythmia detection, recall is more important than precision because it characterizes the classifier's ability to minimize FN in contrast to precision which measures the classifier's ability to minimize FP. F1-score is the harmonic mean of both precision and sensitivity, $F1 = 2 \times \text{PREC} \times \text{SEN}/(\text{PREC} + \text{SEN}) = 2TP/(2TP + FP + FN)$.

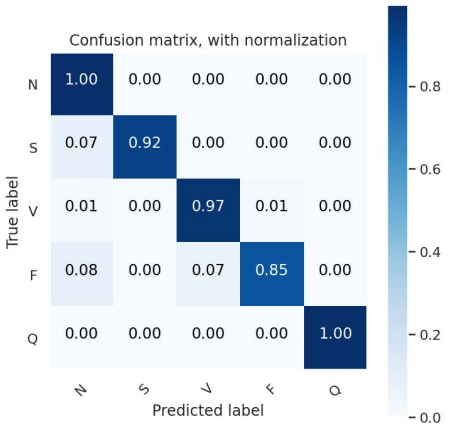


FIGURE 5. Normalized Confusion Matrix of the Conv1D_2D_T model. Numbers inside blocks are the number of samples classified in each class normalized by the total number of samples in the class.

Table 3 shows the per-class accuracy, precision, recall, and F1- scores achieved by the proposed models on the test dataset. The proposed Conv1D_2D_T model can classify the Supraventricular Ectopic and Ventricular Ectopic arrhythmias with 92.49% and 97.5% recall, respectively. In the proposed model, FN result from misclassifying Supraventricular Ectopic beats as normal beats or misclassifying Ventricular Ectopic beats as either normal or fusion beats while FP result from misclassifying fusion beats as Ventricular Ectopic beats as shown by the confusion matrix of Figure 5. FP and FN in the trained models are a direct result of the dataset class

imbalance where the normal beats represent the majority class and the fusion beats represent the minority class. Fortunately, the rate of FP and FN do not exceed 7% and 3%, respectively.

B. MODEL OPTIMIZATION AND TESTING AT THE EDGE

The four selected models are quantized using PTQ and QAT tools. A total of six quantization methods are applied for each model. The applied PTQ methods are float32, float16, dynamic range, full-integer, and int8. The difference between full-integer and int8 quantization is that in the latter both the model and operations input tensors are quantized to int8 while in the former only model operations are quantized to int8. QAT is applied using the Qkeras library which supports quantization of most Keras layers and also supports simultaneous pruning and QAT of TensorFlow models. Unfortunately, the TensorFlow optimization QAT toolkit does not support quantization of the Conv1D layer yet. The models quantized and pruned using Qkeras are denoted as pqs models and are converted to Tflite float32 models after quantization as int8 and pruning 50% of the superfluous weights. Unfortunately, the Tflite library does not support direct quantization of Qkeras models as int8 models without reapplying QAT optimizations which leads to a significant loss of the pqs model performance and is thus excluded.

The developed Tflite models are exported to the raspberry-pi edge device for testing and benchmarking. Two methods are used for testing the Tflite inference engine on the edge device: first, a custom python script is developed for predicting the full test dataset using the developed Tflite models and computes the model accuracy scores and performance metrics; second, the C++ Tflite benchmark tool developed by Google is used to test the performance of the Tflite models on a randomly generated input tensors. It calculates statistics for the model inference time at steady-state and the overall memory usage. The accuracy metrics measured using the python script are accuracy, F1-, recall, precision, and ROC-AUC scores; and the performance metrics measured are: the average inference time, throughput as the number of ECG instances inferred per second, overall memory usage, and model size. The average inference time is calculated by measuring the whole test dataset inference time and dividing it by the number of instances in the dataset while the throughput is calculated as the inverse of the average inference time.

Figure 6 depicts the testing and benchmarking results of the exported Tflite models on the raspberry-pi edge device. As shown by 6(a), Tflite models suffer loss of accuracy metrics compared to the Tensorflow base models. The Conv1D_2D_T Tflite models still achieve the best accuracy scores of above 99% accuracy and 94% F1-score. However, the dynamic range Conv1D_2D_T and Conv1D_2D_T_FIR tflte models suffer from a significant loss of accuracy and fail to achieve more than 40% score which can be attributed to changing the dynamic range of model parameters from the Conv1D to Conv2D layers. The int8 QAT and PTQ Conv1D_2D_T and QAT Conv1D_2D_T_FIR Tflite models lose around 1% of accuracy and %4 of F1-score. On the other hand, the Conv1D_T model losses around 1.5% of accuracy and 6% of F1-score which gives another advantage to the Conv1D_2D_T model. The same conclusions also apply to the recall and ROC-AUC scores.

At the performance level, the Conv1D_T_FIR and Conv1D_T models provide the best results as shown by Figure 6(b). The int8 Conv1D_T model achieves an average inference time of 1.43 ms as executed by the Python script and 0.23 ms as executed by the tflit benchmark tool. The Conv1D_2D_T model achieves a Python and C++ average inference time of 9.2 ms and 7.5 ms, respectively, yet it can be still deployed on the edge device to classify ECG signals in real-time because this inference time is much smaller than the inter-segment RR intervals. The int8 Tflite models have the lowest model size and memory usage. The smallest Tflite model is the Conv1D_T_FIR model of 24.41 KB size and the largest Tflite model is the Conv1D_2D_T model of 89.88 KB size. The overall inference memory usage ranges from 7.01 to 12.59 MB using the Python script and 2.33 to 5.38 MB using the C++ Tflite benchmark tool. The difference between the Python script and C++ Tflite benchmark results is attributed to the Python interpreter overhead. The achieved model sizes and memory usages enable running the Tflite models on a wide range of edge devices with very tight constraints.

C. VISUALIZATION OF THE Conv1D FIR LAYER ACTIVATIONS AND HEATMAP

Since 2013, a wide range of techniques has been developed for visualizing and interpreting 2D CNN activations, filters, and heatmaps [39, Ch. 5]. In Figure 3, the Conv1D filters are visualized in both the time and frequency domains for the Trainable parameter switched from False to True. Herein, we aim to visualize the Conv1D CNN activations and heatmaps. To visualize the FIR Conv1D layer output, the Conv1D_T_FIR model outputs from the Conv1D and activation layers are plotted for random samples of True and False predictions for the five classes in the dataset. The Conv1D_T_FIR model used for visualization has $N_F = 16$, $N_s = 64$, and $N_k = 32$ which is different than the selected best model to enhance the resolution of visualized images. To extract the feature maps, a Keras model is created that takes ECG signals and RR intervals as input tensors and outputs the activations of the Conv1D and the Tanh activation layers tensors [39, Ch. 5]. The ECG signal samples are visualized in the time and frequency domains by plotting the signal amplitude versus time, plotting the signal spectrum obtained using Fast Fourier Transform (FFT), and plotting the signal STFT spectrogram as shown in Figure 7.

The last two rows of each figure illustrate the Conv1D and Tanh activation layer heatmaps displayed as 2D mesh plots in which the horizontal axis is the time axis and the vertical axis is the number of the feature map representing the frequency axis for the designed non-trained FIR Conv1D adjacent BPFs.

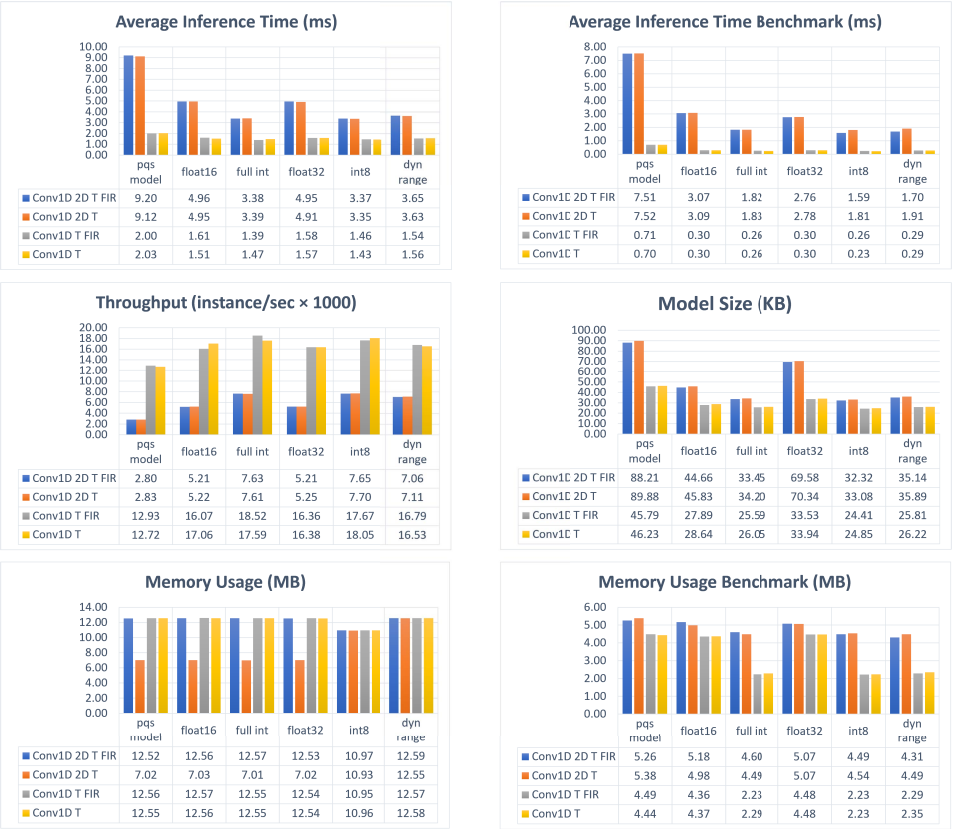
	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	98.53	98.77	98.74	98.77	98.19	24.36
Conv1D 2D T	99.01	99.08	99.07	99.08	98.15	40.04
Conv1D T FIR	98.75	98.65	98.38	98.65	96.25	98.62
Conv1D T	98.80	98.78	98.85	98.78	97.50	98.79

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	89.29	91.33	90.64	91.34	90.27	34.01
Conv1D 2D T	91.54	94.92	94.33	94.92	91.05	33.50
Conv1D T FIR	90.98	93.02	87.55	93.02	77.46	92.82
Conv1D T	91.44	94.62	94.25	94.59	88.55	94.48

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	99.30	99.54	97.48	99.54	96.98	68.48
Conv1D 2D T	99.65	99.74	98.65	99.74	96.86	70.37
Conv1D T FIR	99.32	99.31	97.60	99.31	90.94	99.31
Conv1D T	99.42	99.52	98.54	99.52	95.32	99.52

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	91.23	92.31	92.34	92.31	90.41	18.33
Conv1D 2D T	93.57	94.34	94.11	94.34	90.00	20.15
Conv1D T FIR	92.16	92.18	91.21	92.18	83.12	91.96
Conv1D T	92.54	93.23	93.31	93.21	87.49	93.20

(a) Accuracy, F1-, recall, and ROC-AUC scores



	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	9.20	4.96	3.38	4.95	3.37	3.65
Conv1D 2D T	9.12	4.95	3.39	4.91	3.35	3.63
Conv1D T FIR	2.00	1.61	1.39	1.58	1.46	1.54
Conv1D T	2.03	1.51	1.47	1.57	1.43	1.56

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	7.51	3.07	1.82	2.76	1.59	1.70
Conv1D 2D T	7.52	3.09	1.83	2.78	1.81	1.91
Conv1D T FIR	0.71	0.30	0.26	0.30	0.26	0.29
Conv1D T	0.70	0.30	0.26	0.30	0.23	0.29

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	2.80	5.21	7.63	5.21	7.65	7.06
Conv1D 2D T	2.83	5.22	7.61	5.25	7.70	7.11
Conv1D T FIR	12.93	16.07	18.52	16.36	17.67	16.79
Conv1D T	12.72	17.06	17.59	16.38	18.05	16.53

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	88.21	44.66	33.45	69.58	32.32	35.14
Conv1D 2D T	89.88	45.83	34.20	70.34	33.08	35.89
Conv1D T FIR	45.79	27.89	25.59	33.53	24.41	25.81
Conv1D T	46.23	28.64	26.05	33.94	24.85	26.22

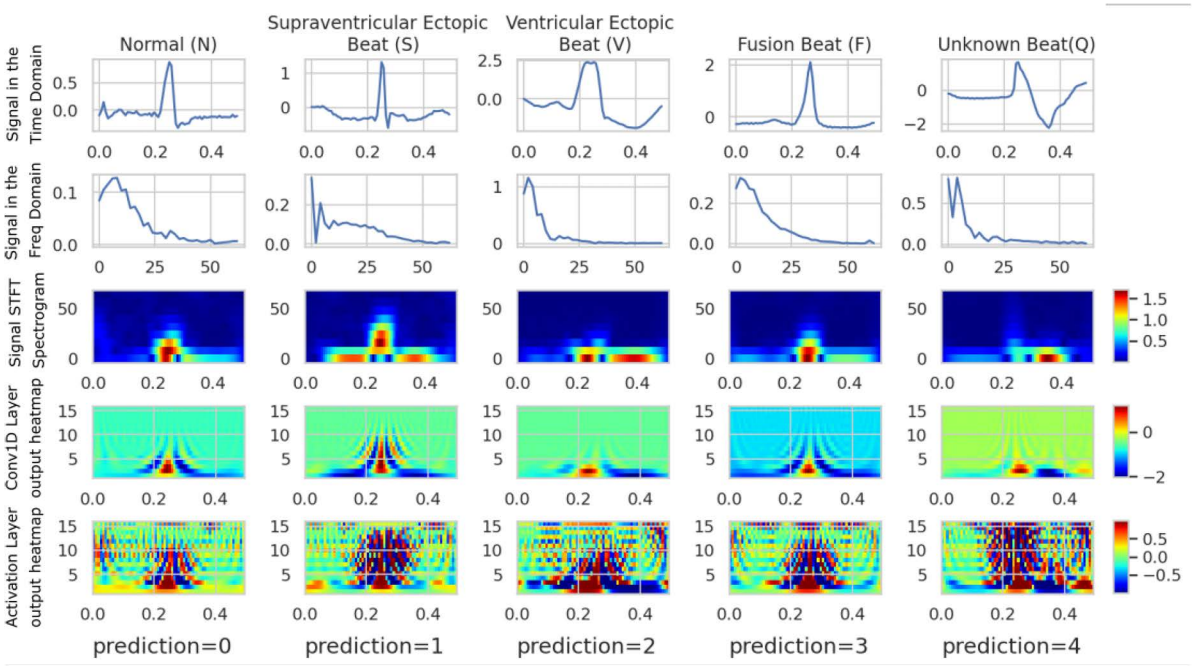
	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	12.52	12.56	12.57	12.53	10.97	12.59
Conv1D 2D T	7.02	7.03	7.01	7.02	10.93	12.55
Conv1D T FIR	12.56	12.57	12.55	12.54	10.95	12.57
Conv1D T	12.55	12.56	12.55	12.54	10.96	12.58

	pqs model	float16	full int	float32	int8	dyn range
Conv1D 2D T FIR	5.26	5.18	4.60	5.07	4.49	4.31
Conv1D 2D T	5.38	4.98	4.49	5.07	4.54	4.49
Conv1D T FIR	4.49	4.36	2.23	4.48	2.23	2.29
Conv1D T	4.44	4.37	2.29	4.48	2.23	2.35

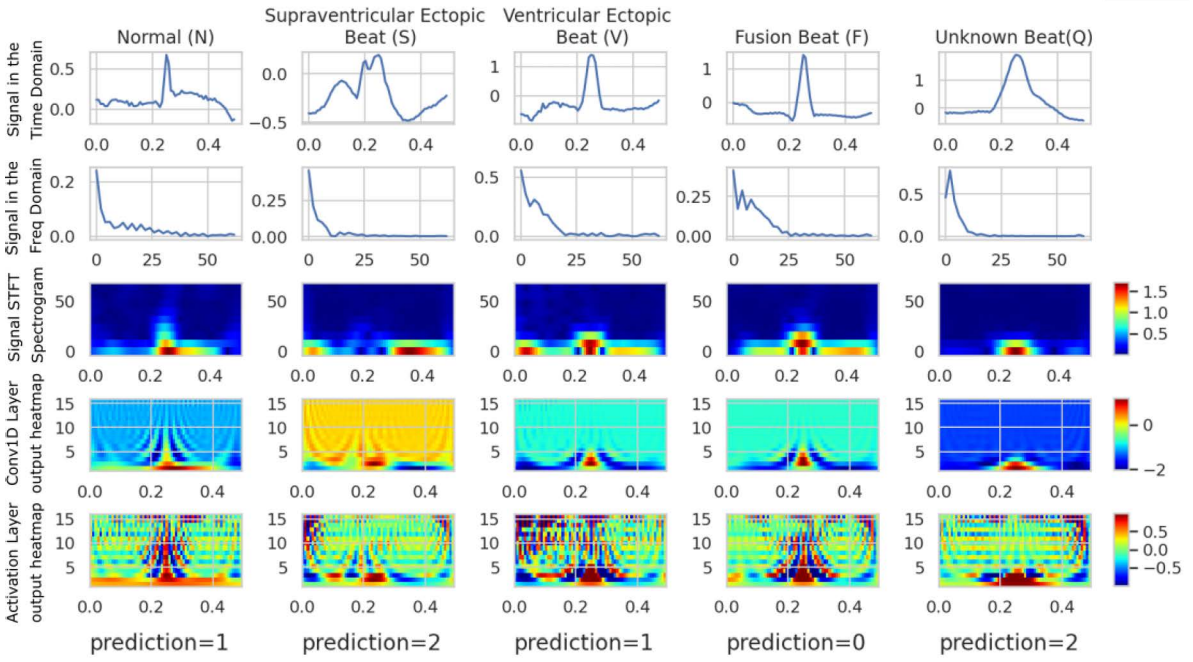
(b) Inference time, throughput, model size, and memory usage

FIGURE 6. **Testing and benchmarking results of the Tflite models on the Raspberry-pi edge device.**

Comparing the STFT and the Conv1D heatmap plots demonstrates that the developed FIR Conv1D layer successfully mimics the STFT algorithm in extracting the time-frequency spectrogram of the ECG signal. The difference between the STFT and the FIR images is that STFT extracts the spectrogram using the Fourier transform algorithm to find the frequency spectrum of the signal while the FIR filter depends on convolution with the FIR BPF filter bank to indicate the existence of specific frequency components at specific time instances. In other words, the STFT spectrogram shows the



(a) Visualization of hit samples



(b) Visualization of miss samples

FIGURE 7. **Visualizing samples of True and False predictions of the Conv1D_2D_T_FIR models in the time and frequency domains and the Conv1D layer heatmaps and Tanh activations. The heatmap size =$N_s \times N_f = 64 \times 16$.**

signal frequency spectrum obtained using FT versus time as a 2D plot while the Conv1D heatmap depicts the BPF filter outputs in the time domain grouped in ascending order of the BPF center frequencies. Instantiating the Tanh activation layer on the top of the Conv1D layer is one of the Keras Tuner step insights which shows to give better classification results compared to using the Relu or Sigmoid activation functions or not using activation functions at all. Enabling training of the Conv1D layer in the Conv1D_2D_T and Conv1D_T models modifies the FIR filter weights and frequency response as shown in Figure 3 which consequently affects the heatmap output from the Conv1D layer. Training the Conv1D input layer weights initialized with the FIR filter coefficients tunes up the frequency response of the filter bank to reduce the

TABLE 4. Comparison between the proposed Conv1D_2D_T model and state-of-the-art ECG classification methods.

Work	Features	Approach	ACC	N			S			V			Average		
			%	PREC	SEN	F1	PREC	SEN	F1	PREC	SEN	F1	PREC	SEN	F1
Proposed model	ECG Segment + RR	Conv1D + Conv2D	99.08	99.62	99.52	99.57	90.52	92.49	91.49	97.88	97.5	97.69	99.09	99.08	99.09
Li et al. [18], 2021	Morph + DWT+RR	IBL	99	99.7	99.35	99.52	90.7	100	95.12	98.4	84.5	90.92	68	-	-
Cui et al. [10], 2021	Conv1D+ DWT+PCA	SVM-RBF	98.35	98.8	98.76	98.78	98.25	99.03	98.64	98.31	96.83	97.56	98.312	98.34	98.36
Liu et al. [19], 2020	WST + PCA	PNN + KNN	99.3	98.5	98.8	98.65	92.7	96.9	94.75	96.4	91.4	93.83	99.6	99.5	99.55
Mousavi and Afghah [20], 2019	ECG Segment + RR	Conv1D + BiLSTM Autoencoder	99.92	99.86	100	99.93	100	96.48	98.21	99.79	99.5	99.64	74.5125	73.915	74.21
Raj and Ray [6], 2018	DOST	ABC + LSTSVM	96.08	98.38	98.11	98.24	57.8	65.34	61.35	88.3	86.93	87.61	96.08	96.08	96.08
Acharya et al. [21], 2017	ECG Segment + DWT	Conv1D	94.03	87.43	91.54	89.44	94.3	90.59	92.41	95.3	94.22	94.76	97.86	96.71	97.28

model loss and enhance its classification accuracy as shown in Figure 3. The center frequencies of the FIR filters are not ordered incrementally as in the non-trainable Conv1D layer and, consequently, their output heatmap has been varied from the heatmaps of Figures 7. Nevertheless, enabling the training of the Conv1D layer proves to give better accuracy scores compared to the non-trained FIR-Conv1D models as illustrated in Table 2.

D. COMPARISON WITH RELATED WORK

In the following, we compare the proposed Conv1D_2D_T model with the state-of-the-art ECG classification methods. The comparison is limited to recent single-lead ECG classification methods applied to the MIT-BIH dataset, categorized according to the AAMI standard, and trained and tested using the random dataset division method to provide a fair comparison. In our comparison, we will compare both the model accuracy metrics and suitability for edge inference which is the main objective of our work. Table 4 depicts the weighted average scores as well as per-class precision, recall, and F1- scores excluding the ''F'' and ''Q'' classes to fully characterize the performance of the compared models. All models listed in this table have been introduced in Section II. Unfortunately, model complexity and run-time performance results are not reported in most related works, however, they can be inferred from the preprocessing and feature extraction stages, model topology, and other model parameters.

The model proposed by Cui *et al.* [10] achieves an average accuracy of 98.35% and per-class precision and recall scores of more than 98%. The authors claim that the proposed model can be used for real-time ECG monitoring but do not provide supportive evidence. Despite achieving good accuracy scores, the DWT preprocessing and feature extraction stages are computationally intensive limiting the model's suitability for edge inference. The model proposed by Li *et al.* [18] achieves an 84.50% recall score of the class ''V'' which is less than most compared works. The proposed method uses a DWT preprocessing stage limiting its applicability for edge inference. The best classification results of the model proposed by Liu *et al.* [19] work are achieved by KNN. However, the testing results are reported for 10-fold cross-validation experiments, not on a separate hold-out test dataset which does not demonstrate the model generalization power. Moreover, the proposed method uses the SWT feature extraction stage which limits its applicability for edge inference.

The model proposed by Mousavi and Afghah [20] is tested for both intra- and inter-patient schemes and the reported scores are superior. Surprisingly, unlike all related works, this model achieves such results without using the RR intervals, which are essential features for ECG classification, raising serious concerns about the presented results. The model has a size of 5.5 MB and it requires neither computationally-intensive preprocessing nor feature extraction stages. Compared to our Conv1D_2D_T model with a maximum model size of 300 KB (non-optimized) and 90 KB for the optimized edge model, the model size is much larger which also indicates that the model inference time and memory usage will be much greater than our model.

The method proposed by Raj and Ray [6] is prototyped on an ARM9 embedded platform and experimentally validated on the MIT-BIH arrhythmia database for both intra- and inter-patient dataset division schemes. The implemented platform is recommended for utilization in hospitals to analyze the long-term ECG recordings however the model size, memory usage, and performance results are not reported. Moreover, the recall and precision metrics for the classes ''S'' and ''V'' are inferior to the model rivals including ours. The accuracy, precision, and recall results of the model proposed by Acharya *et al.* [21] are acceptable but not the best compared to the model rivals including our model. Nevertheless, the proposed model is a good fit for edge deployment due to the minimum preprocessing steps applied excluding the DWT noise removal step.

Eventually, the proposed model outperforms all compared works in terms of the model complexity and computational cost and achieves comparable accuracy results. The achieved results of the proposed classifier enable its deployment on a wide range of edge devices for arrhythmia detection in real- time. Limitations of the proposed classifier include algorithmic and computational limitations. The proposed model is based on STFT and, consequently, it suffers from the STFT time-frequency resolution trade-off. Such a limitation can be overcome by implementing a CWT Conv1D layer to extract high-resolution time-frequency scalogram images. At the computation complexity level, the proposed model is limited in terms of the classifier input size (ECG segment length) and the number of layers (model depth) to meet the edge inference requirements. The proposed model is well suited for ECG classification on a beat-by-beat basis, which is commonly applied in the ECG monitoring and classification literature, rather than long-term ECG segments.

VII. CONCLUSION AND FUTURE WORK

In conclusion, we proposed a novel method for ECG classification optimized for edge deployment and can be embedded in a wearable device for arrhythmia detection. We presented a clear interpretation supported by visualizations of the Conv1D layer operation as an FIR filter and exploited this interpretation to develop a self-contained STFT ECG classifier. The real-time performance of the proposed model has been planned in advance to fit the resource constraints of edge computing. The proposed model is extensively evaluated and benchmarked on a raspberry-pi edge device and the results are reported and discussed. A trade-off is made between the model classification performance and the model complexity and real-time inference performance and the developed models exhibit a good balance between both metrics. The proposed edge model achieves superior real-time performance and computational complexity results and comparable classification accuracy results. A discriminative feature of the proposed model is that it can be readily deployed for real-time ECG monitoring and arrhythmia detection using resource-constrained edge devices.

As future work, we will attempt to improve the model classification performance by designing a self-contained CWT-based CNN. We also plan to extend our work to address the inter-patient division scheme of the MIT-BIH dataset in which the model is trained and tested on heartbeats belonging to different individuals to test the model's capability to generalize and capture inter-individual variations. Moreover, we plan to test the developed model on other internationally recognized ECG databases to investigate the model's generalization capabilities and suitability for practical deployment. Finally, the developed model will be investigated for other relevant time-series classification problems.

REFERENCES

[1] *Cardiovascular Diseases (CVDs)*. Accessed: 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] P. R. Kowey and D. Z. Kocovic, "Ambulatory electrocardiographic recording," *Circulation*, vol. 108, no. 5, pp. e31–e33, Aug. 2003, doi: 10.1161/01.CIR.0000082930.04238.8C.

[3] E. J. D. S. Luz, W. R. Schwartz, G. Cámara-Chávez, and D. Menotti, "ECG-based heartbeat classification for arrhythmia detection: A survey," *Comput. Methods Programs Biomed.*, vol. 127, pp. 144–164, Apr. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0169260715003314

[4] M. Wasimuddin, K. Elleithy, A.-S. Abuzneid, M. Faezipour, and O. Abuzaghleh, "Stages-based ECG signal analysis from traditional signal processing to machine learning approaches: A survey," *IEEE Access*, vol. 8, pp. 177782–177803, 2020.

[5] S. K. Berkaya, A. K. Uysal, E. S. Gunal, S. Ergin, S. Gunal, and M. B. Gulmezoglu, "A survey on ECG analysis," *Biomed. Signal Process. Control*, vol. 43, pp. 216–235, May 2018.

[6] S. Raj and K. C. Ray, "A personalized arrhythmia monitoring platform," *Sci. Rep.*, vol. 8, no. 1, p. 11395, Dec. 2018. [Online]. Available: http://www.nature.com/articles/s41598-018-29690-2

[7] P. S. Addison, "Wavelet transforms and the ECG: A review," *Physiol. Meas.*, vol. 26, no. 5, pp. R155–R199, Oct. 2005.

[8] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A review on deep learning methods for ECG arrhythmia classification," *Expert Syst. Appl., X*, vol. 7, Sep. 2020, Art. no. 100033.

[9] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun, "Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review," *Comput. Biol. Med.*, vol. 122, Jul. 2020, Art. no. 103801.

[10] J. Cui, L. Wang, X. He, V. H. C. De Albuquerque, S. A. AlQahtani, and M. M. Hassan, "Deep learning-based multidimensional feature fusion for classification of ECG arrhythmia," *Neural Comput. Appl.*, pp. 1–15, Sep. 2021.

[11] T. Wang, C. Lu, Y. Sun, M. Yang, C. Liu, and C. Ou, "Automatic ECG classification using continuous wavelet transform and convolutional neural network," *Entropy*, vol. 23, no. 1, p. 119, Jan. 2021.

[12] J. Zhang, A. Liu, D. Liang, X. Chen, and M. Gao, "Interpatient ECG heartbeat classification with an adversarial convolutional neural network," *J. Healthcare Eng.*, vol. 2021, May 2021, Art. no. 9946596.

[13] Z. Wu, T. Lan, C. Yang, and Z. Nie, "A novel method to detect multiple arrhythmias based on time-frequency analysis and convolutional neural networks," *IEEE Access*, vol. 7, pp. 170820–170830, 2019.

[14] A. M. Alqudah, S. Qazan, L. Al-Ebbini, H. Alquran, and I. A. Qasmieh, "ECG heartbeat arrhythmias classification: A comparison study between different types of spectrum representation and convolutional neural networks architectures," *J. Ambient Intell. Humanized Comput.*, pp. 1–31, Apr. 2021. [Online]. Available: https://link.springer.com/article/10.1007/s12652-021-03247-0#article-info

[15] M. Cao, T. Zhao, Y. Li, W. Zhang, P. Benharash, and R. Ramezani, "ECG heartbeat classification using deep transfer learning with convolutional neural network and STFT technique," 2022, *arXiv:2206.14200*.

[16] J. Huang, B. Chen, B. Yao, and W. He, "ECG arrhythmia classification using STFT-based spectrogram and convolutional neural network," *IEEE access*, vol. 7, pp. 92871–92880, 2019.

[17] A. Ullah, S. M. Anwar, M. Bilal, and R. M. Mehmood, "Classification of arrhythmia by using deep learning with 2-D ECG spectral image representation," *Remote Sens.*, vol. 12, no. 10, p. 1685, Apr. 2020.

[18] J. Li, Y. Zhang, L. Gao, and X. Li, "Arrhythmia classification using biased dropout and morphology-rhythm feature with incremental broad learning," *IEEE Access*, vol. 9, pp. 66132–66140, 2021.

[19] Z. Liu, G. Yao, Q. Zhang, J. Zhang, and X. Zeng, "Wavelet scattering transform for ECG beat classification," *Comput. Math. Methods Med.*, vol. 2020, pp. 1–11, Oct. 2020.

[20] S. Mousavi and F. Afghah, "Inter- and intra- patient ECG heartbeat classification for arrhythmia detection: A sequence to sequence deep learning approach," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 1308–1312.

[21] U. R. Acharya, "A deep convolutional neural network model to classify heartbeats," *Comput. Biol. Med.*, vol. 89, pp. 389–396, Oct. 2017.

[22] G. B. Moody and R. G. Mark, "The MIT-BIH arrhythmia database on CD-ROM and software for use with it," in *Proc. Comput. Cardiol.*, 1990, pp. 185–188.

[23] 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, Jun. 2001.

[24] Association for the Advancement of Medical Instrumentation. (Jan. 2012). *AMI EC57—Testing and Reporting Performance Results of Cardiac Rhythm and ST Segment Measurement Algorithms Engineering360*. [Online]. Available: https://standards.globalspec.com/std/14370017/AAMI%20EC57
[25] C.-C. Lin and C.-M. Yang, "Heartbeat classification using normalized RR intervals and morphological features," *Math. Problems Eng.*, vol. 2014, pp. 1–11, May 2014. [Online]. Available: http://www.hindawi.com/journals/mpe/2014/712474/
[26] 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.
[27] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, Apr. 2021, Art. no. 107398. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327020307846
[28] A. V. Oppenheim, A. S. Willsky, S. H. Nawab, and G. M. Hernández, *Signals & System*. London, U.K.: Pearson, 1997.
[29] L. Stankovic and D. Mandic, "Convolutional neural networks demystified: A matched filtering perspective based tutorial," 2021, *arXiv:2108.11663*.
[30] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, p. 27, Dec. 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0192-5
[31] *Quantization Aware Training With TensorFlow Model Optimization Toolkit—Performance With Accuracy*. Accessed: 2022. [Online]. Available: https://blog.tensorflow.org/2020/04/quantization-aware-training-with-tensorflow-model-optimization-toolkit.html
[32] *TensorFlow Lite: ML for Mobile and Edge Devices*. Accessed: 2022. [Online]. Available: https://www.tensorflow.org/lite/
[33] C. N. Coelho, Jr., A. Kuusela, H. Zhuang, T. Aarrestad, V. Loncar, J. Ngadiuba, M. Pierini, and S. Summers, "Ultra low-latency, low-area inference accelerators using heterogeneous deep quantization with QKeras and hls4ml," 2020, *arXiv:2006.10159v1*.
[34] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017, *arXiv:1712.05877*.
[35] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*.
[36] *TensorFlow Model Optimization Toolkit—Post-Training Integer Quantization*. Accessed: 2022. [Online]. Available: https://blog.tensorflow.org/2019/06/tensorflow-integer-quantization.html
[37] *Gradient Paperspace*. Accessed: 2022. [Online]. Available: https://gradient.run/
[38] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," 2017, *arXiv:1707.01836*.
[39] F. Chollet, *Deep Learning With Python*. New York, NY, USA: Simon and Schuster, 2021.

MOHAMMED M. FARAG (Member, IEEE) was born in Egypt. He received the B.S. and M.S. degrees in electrical engineering from Alexandria University, Egypt, in 2003 and 2007, respectively, and the Ph.D. degree in computer engineering from Virginia Tech Institute and State University, in 2012.

He worked as a Teaching Assistant at Alexandria University, from 2003 to 2009. From 2009 to 2013, he worked as a Research Assistant at Virginia Tech University. From 2013 to 2017, he joined the Electrical Engineering Department, Alexandria University, as an Assistant Professor. In 2018, he started secondment leave to work at the Electrical Engineering Department, College of Engineering, King Faisal University, Saudi Arabia. His research interests include machine learning, network-on-chip, system-on-chip design, hardware-based design, FPGA prototyping, and cyber-physical security.