Research Article/Electrical, Electronics, and Telecommunications Engineering

http://doi.org/10.15446/ing.investig.114530

Lightweight Deep Learning for Atrial Fibrillation Detection: Efficient Models for Wearable Devices

Aprendizaje profundo ligero para la detección de fibrilación auricular: modelos eficientes para dispositivos portátiles

Carlos A. Fajardo[1], Andrés S. Parra[2], and Tania Castellanos-Parada[3]

ABSTRACT

The timely and accurate detection of atrial fibrillation (AF) is crucial for an early intervention and for reducing the associated health risks. Wearable technology has emerged as a viable solution for continuous AF monitoring, but deploying accurate AF detection models on resource-constrained devices remains a challenge due to the high computational and memory demands. This study proposes a lightweight and efficient deep learning approach for real-time AF diagnosis in portable devices. We designed a series of convolutional neural network (CNN) models optimized for high accuracy while maintaining a minimal computational footprint. To further enhance efficiency, we explored deep learning compression techniques, including pruning, quantization, and knowledge distillation. Our results demonstrate that the proposed models achieve state-of-the-art accuracy while significantly reducing memory usage and computational complexity, making them suitable for real-time deployment. Additionally, we validated their feasibility by implementing them on a microcontroller, showcasing their practicality for wearable applications. This research paves the way for accessible, low-power, and high-accuracy AF detection in real-world settings, enabling early diagnosis and timely medical intervention without the need for continuous clinical supervision.

Keywords: cardiac arrhythmia, deep learning, detection, ECG, electrocardiogram, machine learning, portable device

RESUMEN

La detección temprana y precisa de la fibrilación auricular (FA) es fundamental para una intervención oportuna y la reducción de riesgos asociados. La tecnología *wearable* ha surgido como una solución viable para el monitoreo continuo de la FA, pero la implementación de modelos precisos de detección en dispositivos con recursos limitados sigue siendo un desafío debido a las altas demandas computacionales y de memoria. Este estudio propone un enfoque de aprendizaje profundo ligero y eficiente para el diagnóstico en tiempo real de la FA en dispositivos portátiles. Diseñamos una serie de modelos de redes neuronales convolucionales (CNN) optimizados para alcanzar alta precisión a la vez que mantenían un bajo costo computacional. Para mejorar aún más la eficiencia, exploramos técnicas de compresión de aprendizaje profundo, incluyendo poda, cuantización y destilación de conocimiento. Nuestros resultados demuestran que los modelos propuestos logran una precisión de última generación mientras reducen significativamente el uso de memoria y la complejidad computacional, lo que los hace adecuados para su implementación en dispositivos de borde. Además, validamos su viabilidad implementándolos en un microcontrolador, demostrando su su aplicabilidad en *wearables*. Esta investigación abre el camino para una detección de FA accesible, de bajo consumo y de alta precisión en entornos reales, permitiendo un diagnóstico temprano y una intervención médica oportuna sin necesidad de supervisión clínica continua.

Palabras clave: arritmia cardiaca, aprendizaje profundo, detección, ECG, electrocardiograma, aprendizaje automático, dispositivo portátil
Received: May 20th 2025
Accepted: March 31st 2025

Introduction

Atrial fibrillation (AF) is the most common arrhythmia diagnosed in clinical practice, and its widespread prevalence is considered alarming, with some researchers predicting an epidemic within the next 10 to 20 years [1].

The primary method for detecting AF is analyzing the heart's electrical activity via an electrocardiogram (ECG), which captures patterns that reveal heart function and potential abnormalities. Early detection of AF is crucial in reducing morbidity and mortality. However, AF detection is particularly challenging due to its paroxysmal nature, where episodes occur sporadically and are often asymptomatic in early stages. Consequently, long-term monitoring is recommended for patients exhibiting occasional AF-related symptoms.

The systematic diagnosis of paroxysmal atrial fibrillation (PAF) is an important public health concern, as early identification allows for timely intervention with oral anticoagulation and catheter ablation, treatments that can be curative when applied at the appropriate time [2]. In recent years, machine learning-based computer-aided diagnosis (CAD) systems have demonstrated high accuracy in AF detection [3], significantly improving clinical workflows [4].

Motivation and challenges

Despite these advancements, current deep learning models for AF detection face key limitations that hinder their deployment in wearable devices. Most high-performing

[1]Electronics engineer, Universidad Industrial de Santander, Colombia. PhD, Universidad Industrial de Santander, Colombia. Affiliation: faculty professor, E-mail: cafajar@uis.edu.co
[2]Electronics engineer, Universidad Industrial de Santander, Colombia. Engineer in charge of control and instrumentation reliability. Ecopetrol. Barrancabermeja, Colombia. Email: andres.parraes@ecopetrol.com.co
[3]Electronics engineer, Universidad Industrial de Santander, Colombia. MSc, Universidad Industrial de Santander, Colombia. Affiliation: graduate student

models rely on architectures with millions of parameters, making real-time inference impractical due to high memory and computational demands. Transformer-based and ensemble models, while improving classification accuracy, introduce latency and energy inefficiency, making them unsuitable for edge deployment.

Our approach and contributions

To address these challenges, we propose a lightweight convolutional neural network (CNN) architecture optimized for real-time inference on resource-constrained devices. Our key contributions include:

- **An efficient deep learning model**: We designed a CNN with significantly fewer parameters than traditional models while maintaining state-of-the-art accuracy, making it viable for real-time applications.
- **Model compression strategies**: We explored pruning, quantization, and knowledge distillation to further reduce model size and computational cost while preserving performance.
- **Robustness to noisy signals**: We utilized Icentia11k, one of the largest public ECG datasets, ensuring better generalization. Our approach considers real-world ECG conditions, including the noise and signal variability that are common in wearable devices.
- **Hardware deployment and evaluation**: We implemented our model on a microcontroller, measuring performance in terms of accuracy, memory footprint, inference time, and computational complexity (FLOPs).

This research extends our previous work on low-complexity deep learning models for AF diagnosis [5], incorporating additional model optimizations, the evaluation of a larger dataset, and real-time hardware implementation.

This study follows a structured approach to address the challenges of AF detection in resource-constrained environments. Section 2 presents the dataset used in this research, detailing its characteristics and relevance to the problem at hand. Building upon this foundation, Section 3 introduces the proposed deep learning models, highlighting their design choices and suitability for efficient ECG classification. To further enhance the practicality of our approach, Section 4 explores various compression strategies, including knowledge distillation, pruning, and quantization, with the aim of reducing computational complexity while maintaining classification performance. The implementation of these models on resource-limited hardware is discussed in Section 5, where we evaluate real-time inference feasibility. In Section 6, we present a comprehensive analysis of our results, comparing them against those of related studies and providing key insights into the effectiveness of our method for AF detection. Finally, Section 7 summarizes the study's contributions, discusses its implications, and outlines potential directions for future research.

Data description

Analyzing the heart's electrical activity is the main technique for detecting arrhythmias. This electrical activity is recorded by electrodes on the skin and plotted as a voltage *vs.* time graph (Fig. 1). This graph is called an *electrocardiogram* (ECG).

There are different types of waves in an ECG signal, such as P-waves (atrial depolarization), T- and U-waves (ventricular repolarization), and QRS complexes (ventricles depolarization). These waves are used to identify anomalies in the behavior of the heart, like cardiac arrhythmia [6, 7].

AF is caused by the asynchronous contraction of the atria due to the fast firings of electrical impulses. AF is characterized by the lack of sinus P-waves, irregular and fast ventricular contraction, an irregular and unusual RR interval, and atrial heart rate oscillating between 140 to 160 beats per minute (bpm) [8].

Atrial flutter is another type of cardiac arrhythmia. Unlike AF, during an atrial flutter episode, the electrical activity of the atria is synchronous, and the atrial heart rate oscillates between 250 to 350 bpm.

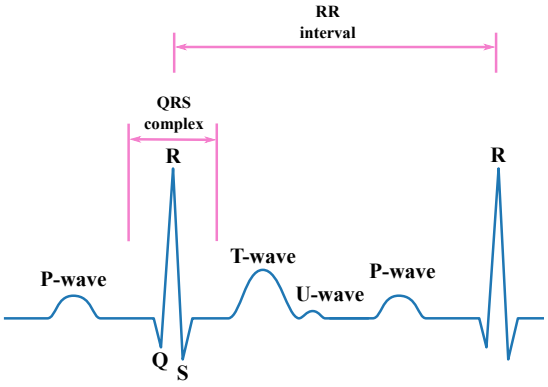


Figure 1. ECG waveform and its main patterns (P-, T-, and U-waves, QRS complexes, and RR intervals)
Source: Authors

In this work, we used the Icentia11k dataset [9], which contains normal sinus rhythms, noises, and AF and atrial flutter signals. As suggested by [4], we merged the AF and atrial flutter signals into one single class.

The Icentia11k dataset was selected for the following reasons. First, it is the largest public ECG dataset, with 11 000 patients and 2 billion labeled beats. This helps to reduce generalization error. Second, this dataset was collected using a single-lead heart monitor device, which allows for a more straightforward future deployment in wearable hardware. Finally, the Icentia11k dataset has a large number of noise signals (around 40%). As previously mentioned, the ability to deal with noise signals is an important goal in the development of wearable devices.

The signals were acquired using the CardioSTAT device [10]. Each patient used the device for a period of two weeks. This device automatically detects the heartbeat. Subsequently, the corresponding labels were verified by a cardiologist who analyzed the complete recording. The dataset has 50 segments per patient, each with 1 048 576 samples. From each of these segments, four frames with 2 049 samples were extracted. Table 1 shows the dataset statistics.

Table 1. Dataset statistics

Statistic	# Units
Number of Patients	11 000
Number of labeled beats	2 774 054 987
Sample Rate	250 Hz
Frame size	2 049 Samples
Total number of frames	2 555 592

Source: Authors

We split the dataset into training, validation, and test sets, using 64%, 16%, and 20% of the data, respectively (Fig. 2). It is important to note that each set has different patients; there are no shared patients between sets.

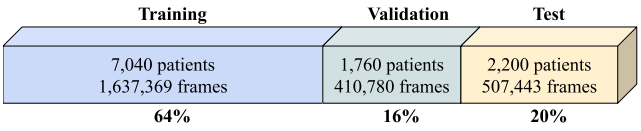


Figure 2. Strategy used to split the data
Source: Authors

Models

This research aimed to empirically find the smallest and most accurate deep learning model. It proposes a CNN inspired by [4], [11], and [12]. To this effect, we tested several configurations by varying the kernel size, depth, and width.

The network architecture consists of N residual blocks (Fig. 3), where N, I_{ch}, and S_j are hyperparameters that allow varying the model size.

The first and last layers of the model are special cases due to this pre-activation block structure [11]. The model uses residual blocks as a means to deal with the vanishing or exploding gradient problem [13]. Moreover, the model includes skip-connections, which favor the propagation of information in deep neural networks, and it uses batch normalization to keep values within bounds and avoid saturation. It applies dropout to prevent overfitting during training, followed by a rectified linear activation unit (ReLU). Each residual block culminates in average pooling. Furthermore, maxpooling layers were also used in the skip connections to maintain dimensional consistency when the two separate paths joined back together at each block. The model finishes with a fully connected layer, followed by the *softmax* function, which has three outputs corresponding to the probability of each class (normal, arrhythmia, or noise).

Compact models

Neural networks are typically oversized [14]. Thus, this research aimed to find the smallest and most accurate model empirically. The goal was to determine the minimum values for the model's kernel size, width, and depth. This process focused on the convolutional layers' number of trainable parameters, as they are the majority in the model.

In the proposed model (Fig. 3), the first layers have the same number of initial output channels (I_{ch}). In the residual blocks, the number of output channels (C^j_{out}) is given by

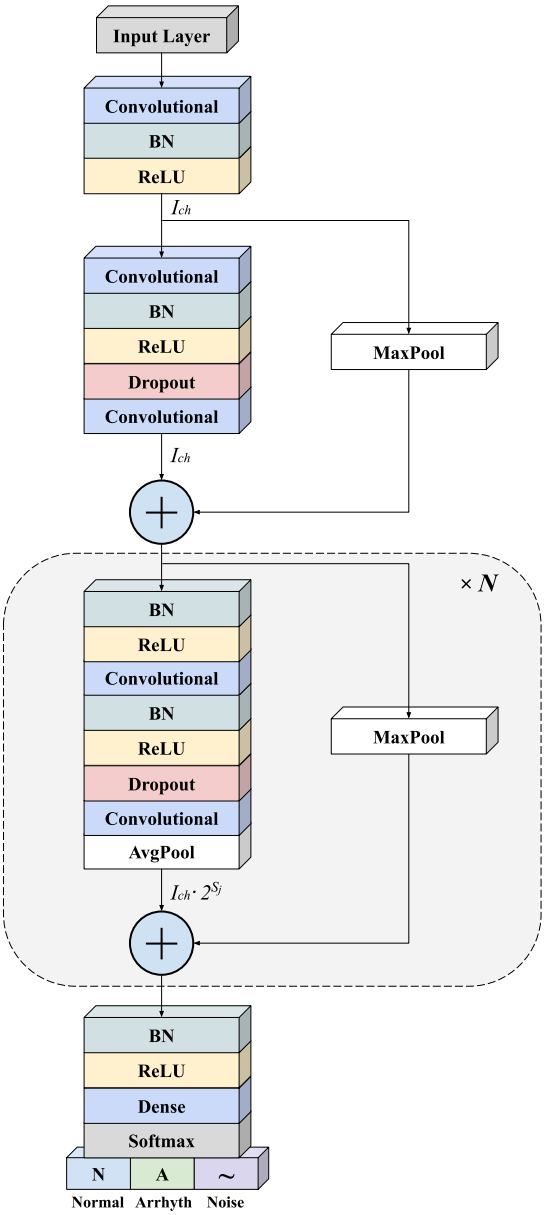


Figure 3. The CNN consists of N residual blocks and accepts raw ECG data as input. It outputs a prediction of normal (N), cardiac arrhythmia (A), or noise signals (~). The number of output channels for the first layers is I_{ch}. The number of output channels for the layers in the residual blocks is $I_{ch} \cdot 2^{s_j}$. The depth and width of the network are changed using the hyperparameters N, I_{ch}, and s_j
Source: Authors

$$C^j_{out} = I_{ch} \cdot 2^{s_j}, \tag{1}$$

where j is the number of the residual block, and s_j and I_{ch} are hyperparameters that allow setting the increment of the output channels throughout the model. This work also applied different strides in the residual blocks.

Let C^j_{in} and C^j_{out} be the input and output channel sizes for the 1D-convolutional layer in the *jth*-residual block (for notation simplicity, assume that the input and output activations have the same spatial size). The convolutional layer will have $C^j_{out} \cdot C^j_{in}$ filters. Thus, for a kernel size of K, the total number of trainable parameters in the 1-D convolutional layer will be

$$P\_conv = K \cdot C_{in}^{j} \cdot C_{out}^{j} + C_{out}^{j}, \quad (2)$$

where C_{out}^{j} and C_{in}^{j} are calculated using Eq. (1), adjusting the corresponding value of s_j. The last C_{out}^{j} corresponds to the bias parameters.

We tested several configurations by varying the model's kernel size (K), depth (N), and width (I_{ch} and s_j). Kernel size, a highly influential hyperparameter, underwent extensive training. Our findings indicate that the best value for kernel size is $K = 16$. We also tested different ways to apply downsampling to the CNN by setting a stride greater than 1. The late downsampling strategy [15] was used, which has been shown to improve accuracy on a limited budget of parameters. This strategy aims to generate large activation maps at the beginning of the network, which can lead to higher classification accuracy.

We trained several models with the aim of determining their F1-score. Fig. 4a illustrates the top-performing models of our study. Notably, with just 73 343 parameters, we achieved an F1-score of 0.909, and larger models did not report a significant improvement. Therefore, we decided to focus on the five smallest models (CNN1-CNN5). Fig. 4b shows the number of floating-point operations (FLOPs) for our five CNN models. This number is a measurement used to assess computational complexity. As expected, the larger models yielded higher FLOPs values.

Fig. 5 provides a visual representation of these five models, showcasing their architectural components. Input and output layers are depicted in gray boxes, while the red boxes indicate special-cased blocks. The blue boxes correspond to the N residual blocks, and their output dimensions are highlighted. As an example, the CNN5 model encompasses two red special-cased blocks, each with four output channels. Furthermore, the model incorporates 13 blue residual blocks, wherein the number of output channels progressively increases every fourth residual block. Note that, as the number of filters grows, the output dimension undergoes subsampling.

Training

The deep learning models were implemented in Python, using the Keras library with a TensorFlow backend. The networks were trained using the Adam optimizer with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and a mini-batch size of 128. The initial learning rate was set to 0.001 and was reduced by a factor of 10 when the loss plateaued. To prevent overfitting, we employed early stopping, terminating training when the validation loss failed to improve for a predefined number of epochs. The model with the lowest validation loss was selected for the final evaluation.

Compressed models

Several strategies to compress deep learning models have been proposed, which aim to reduce memory and computational requirements without significantly compromising accuracy [14]. By applying these compression techniques, deep learning models can be made more compact, making them easier to deploy on edge devices. In this study, we used knowledge distillation, pruning, and quantization as compression methods. The subsequent subsections detail the implementation of these strategies.

Knowledge distillation

Knowledge distillation (KD) is a strategy for model compression, whose foundations were established by [16] and were then improved and generalized [17]. KD aims seeks to transfer knowledge from a well-trained larger model (teacher) to a small one (student). [17] suggested that, when a teacher assigns a relatively high probability to the wrong categories, the teacher can offer valuable information regarding similarities between classes, which can be used to train the student. However, in most cases, the teacher assigns very high probabilities to the correct classes and close-to-zero probabilities to the wrong classes. In these cases, the transferred information from the teacher to the student is very similar to the one-hot ground truth labels. To tackle this problem, [17] introduced the concept of *temperature* (T) to calculate the *softmax* equation as follows:

$$p_i(z_i; T) = \frac{exp(\frac{z_i}{T})}{\sum_{j=1}^{c} exp(\frac{z_j}{T}))}, \quad (3)$$

where z_i are the logits and C is the number of classes. When T is set to a larger value, the probability distribution becomes softer. The soft probabilities calculated using Eq. (3) are known as *soft labels*, whereas the ground-truth labels are *hard labels*.

In KD, the student is trained using two different loss functions. The first one calculates the Kullback-Leibler divergence between the teacher and the student's soft labels. The second function is the standard cross-entropy (L_{ce}), calculated with model predictions and ground-truth hard labels. Thus, in KD, the complete loss function L is a linear combination between L_{kl} and L_{ce}:

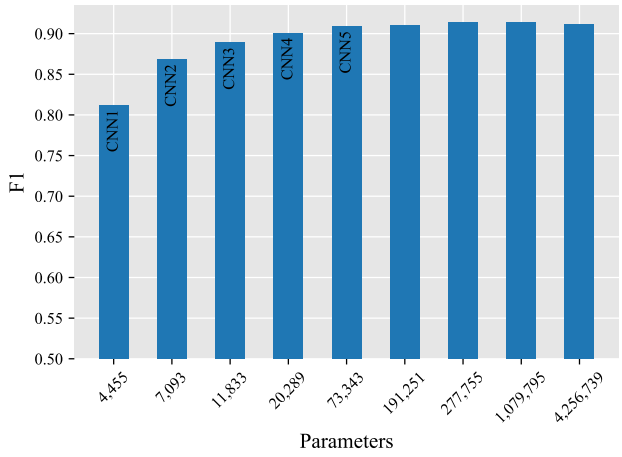
$$L = \alpha L_{ce} + (1 - \alpha) L_{KD}. \quad (4)$$

KD was applied using teachers with as much as 10 times more parameters in order to improve the performance metrics of the student models (CNN1-CNN5). Many tests using different values of α and T, *i.e.*, Eqs. (3) and (4), were carried out, obtaining better results when using high temperatures (> 15) and small α values (< 0.5).

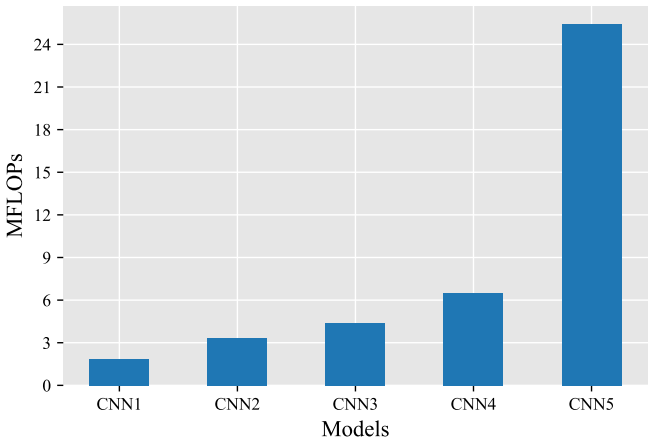
Pruning strategy

Pruning is a compression technique used to reduce the model size by removing non-important connections. [18] proposed a magnitude-based weight pruning strategy to reduce model size and facilitate the deployment of DNNs on wearable and mobile devices. The method prunes network connections in such a way that the original accuracy is preserved. The pruning steps can be consulted in more detail in [18].

We pruned the models using two pruning schedules: zeroing weights constantly throughout the training process or gradually zeroing weights in a polynomial decay fashion. Better results were obtained with the constant pruning schedule. The models CNN1-CNN5 were pruned with 50%



(a) F1-score *vs.* number of parameters



(b) Number of floating-point operations (FLOPs)

Figure 4. F1-scores and FLOPs for the models CNN1-CNN5

Table 2. Evaluation of different compression methods

Model	Baseline		KD		Pruning		Quantization	
	F1	Size (KB)	F1	Size (KB)	F1	Size (KB) / (% reduction)	F1	Size (KB) / (% reduction)
CNN1	0.81	31	0.82	31	0.80	25 / (21%)	0.80	8 / (75%)
CNN2	0.87	48	0.87	48	0.85	38 / (21%)	0.86	12 / (75%)
CNN3	0.89	63	0.90	63	0.89	46 / (26%)	0.88	16 / (75%)
CNN4	0.90	98	0.90	98	0.89	69 / (29%)	0.89	24 / (75%)
CNN5	0.91	296	0.91	296	0.91	193 / (35%)	0.90	74 / (75%)

Source: Authors

sparsity because using a higher value yielded a poor trade-off between F1-score and size. Nevertheless, models larger than CNN5 allowed for pruning with a sparsity greater than 50% without compromising the metrics.

Quantization

This research used an 8-bit quantization implementation based on [19]. Considering an unsigned 8-bit representation, we mapped the floating-point variable to 256 values (0, 255). To this effect, two parameters were considered: scale (S) and zero-point (Z), which allow mapping the floating-point value (r) to the corresponding 8-bit value (q) using the following equation:

$$r = S(q - Z). \tag{5}$$

Results obtained from the compression methods

Table 2 presents the results of applying these three compression strategies to our proposed models. The table shows the F1-score and model size (in KB). The percent reduction in model size, achieved through pruning and quantization, is also presented.

KD did not enable a significant improvement in the F1-score and failed to reduce the model size. Conversely, pruning reduced the model size and, in certain cases, it even improved performance due to the regularization effect of the pruning strategy. Remarkably, quantization emerged as the most effective method, reducing the model size by 75% with only a slight decrease of 0.01 in the F1-score.

Hardware deployment

We evaluated the models by implementing the inference stage in a Teensy 4.1 microcontroller based on ARM Cortex-M7, working at a frequency of up to 600 MHz. It has 8 MB of FLASH memory and 1 MB of RAM [20].

Table 3 presents the performance of our implemented models. It includes the following metrics [21]:

- **Model size** is reported in terms of the number of parameters used.
- **Accuracy** is measured using the F1-score, which assesses the model's performance in terms of precision and recall.
- **Computational cost** is measured in the number of millions of FLOPs required for inference.
- **Peak memory footprint** indicates the maximum amount of memory (in KB) consumed by the models during inference.
- **Inference time** (in ms) is measured at both the minimum and maximum operating frequencies.

The results presented in Table 3 demonstrate the feasibility of implementing the models on this resource-constrained device. The peak memory footprint in the flash memory ranged from 1% (CNN1) to 4% (CNN5) of the available memory. RAM usage ranged from 11% (CNN1) to 53% of the available memory. The inference time was sufficiently small to support real-time inference. However, it is important to

Table 3. Evaluation metrics of the models on the Teensy 4.1

Model	Size # parameters	F1	MFLOPs	Peak Memory Flash (KB)	Peak Memory RAM (KB)	Inference time (ms) 150 MHz	Inference time (ms) 600 MHz
CNN1	4455	0.81	1.84	63.7	115.3	227	58
CNN2	7093	0.87	3.35	77.9	148.1	479	122
CNN3	11 833	0.89	4.39	96.3	213.7	559	145
CNN4	20 289	0.90	6.53	131.1	295.7	816	210
CNN5	73 343	0.91	25.45	344.0	529.0	2039	599

Source: Authors

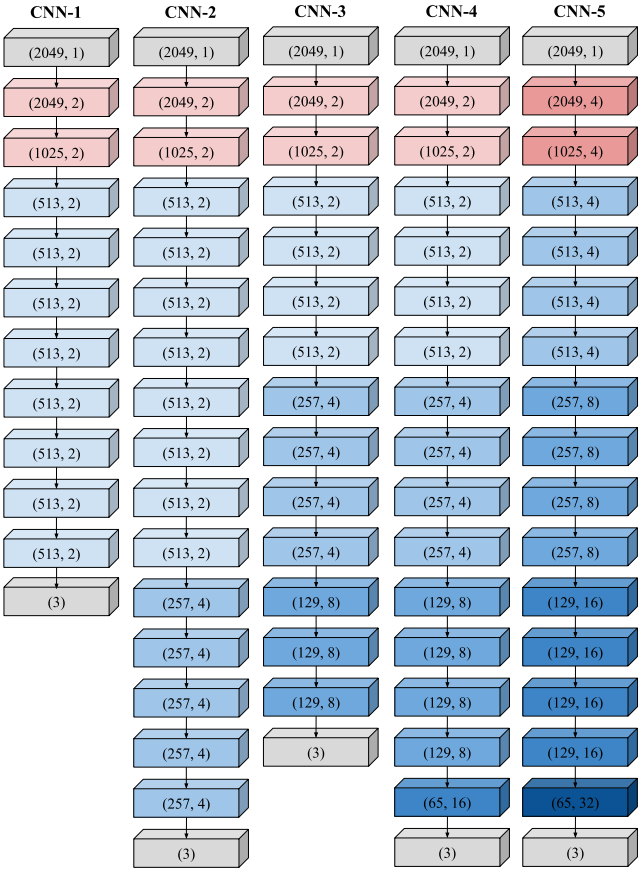


Figure 5. Architecture used in our five models. Each box represents a residual block and its output dimension. Boxes in gray are input/outputs, boxes in red are the special-cased blocks, and boxes in blue represent the N residual blocks.
Source: Authors

note that the RAM requirement may be a critical factor to consider.

Discussion

Among the classical algorithms used for ECG signal classification are support vector machines [26], naive Bayes [27], linear and quadratic discriminant analysis, and random forests [28], among others. However, these classical techniques typically require two steps: feature extraction (pre-processing) and classification. This feature engineering process often relies on domain expertise and can introduce biases. In contrast, deep learning approaches can learn hierarchical features directly from raw signals, eliminating the need for manual feature extraction. Moreover, reducing pre-processing steps can lower computational costs and energy consumption, making deep neural networks more suitable for real-time applications.

Deep learning techniques have significantly advanced ECG classification and arrhythmia detection [29, 4, 30, 31]. CNNs can capture temporal and spatial patterns in ECG signals [23, 32], while recurrent neural networks (RNNs) and transformer-based models enhance the modeling of sequential dependencies [33].

Recent approaches integrate hybrid deep learning models that combine convolutional and recurrent architectures to enhance feature extraction and classification performance [34]. Additionally, temporal convolutional networks (TCNs) have emerged as an alternative to recurrent models, offering greater efficiency in capturing long-term dependencies while reducing computational complexity [35]. Ensemble methods that combine multiple deep learning models have also been explored to improve robustness and generalization [36].

These studies primarily aim to improve classification accuracy, often at the expense of increased model size. However, larger models pose challenges regarding memory consumption and computational complexity [14, 24], which are critical for deployment on resource-constrained edge devices. The inclusion of recurrent layers { such as long short-term memory (LSTM) cells { or attention mechanisms can further elevate computational costs, making real-time implementation impractical [37].

Unlike previous deep learning models that prioritize accuracy at the expense of high computational complexity, our approach balances efficiency and performance. As shown in Table 4, models with a large number of parameters often achieve state-of-the-art accuracy but are impractical for wearable devices due to their excessive memory and power requirements.

Table 4 provides an analysis of our proposed models *vs.* recent ECG classification studies that use large datasets (more than 8000 patients) and explicitly report model parameters. This comparison focuses on key factors such as the size of the dataset, the number of classes, the deep learning techniques used, and the classification performance. Given the challenges of a direct comparison due to variations in the datasets, the number of ECG leads, and model objectives, our analysis prioritizes studies that align with our focus on both accuracy and computational efficiency. Specifically, we included research that reports on model parameters, allowing for estimations of computational cost, a crucial factor for deployment on wearable devices. Furthermore, we prioritized studies

Table 4. Recent studies, on larger data sets, classifying ECG signals using Deep Learning techniques.

Study	Database	Total data & classes	Number of patients	Technique	Number of parameters	Results
[4]	Zio Monitor	91 232 records, 12 classes	53 549	CNN	10 473 635	F1 = 0.837
[22]	PhysioNet Challenge 2017	8528 records, 4 classes	8528	CNN	262 344	F1 = 0.820
[23]	1st China Physiological Signal Challenge	9831 records, 8 classes	9831	ATI-CNN	4 984 640	F1 = 0.812
[24]	Icentia11k	550 000 records, 4 classes	11 000	Contrastive CNN (few-shot)	250 000	F1 = 0.801*
[25]	ECG5000	5000 records	Not specified	Temporal CNN (TCN)	10 200	F1 = 0.8413
CNN1 (**Ours**)	Icentia11k	550 000 records, 4 classes	11 000	CNN	4455	F1 = 0.813
CNN2 (**Ours**)	Icentia11k	550 000 records, 4 classes	11 000	CNN	7093	F1 = 0.872
CNN3 (**Ours**)	Icentia11k	550 000 records, 4 classes	11 000	CNN	11 833	F1 = 0.891
CNN4 (**Ours**)	Icentia11k	550 000 records, 4 classes	11 000	CNN	20 289	F1 = 0.902
CNN5 (**Ours**)	Icentia11k	550 000 records, 4 classes	11 000	CNN	73 343	F1 = 0.909

Source: Authors

reporting the F1-score, as accuracy alone can be misleading in unbalanced datasets.

A notable trend in Table 4 is that models trained on larger datasets require more parameters for effective generalization. Our study used one of the largest publicly available ECG datasets in terms of both signal count and patient coverage. Remarkably, our CNN-4 model, with only 20 289 parameters, achieved an F1-score of 0.902, demonstrating high efficiency relative to its size. Furthermore, its 8-bit quantized version reduced the memory footprint by 75% while incurring a minimal F1-score reduction of 0.01. This result underscores the balance between model performance and deployment feasibility, and therefore the importance of designing efficient architectures for real-world applications.

Future work should focus on optimizing deep learning architectures to further reduce computational complexity while maintaining high accuracy. Lightweight models, KD [38], and hardware-aware optimizations are promising directions to bridge the gap between accuracy and efficiency in ECG classification.

Conclusions and future work

This study addressed the challenges of atrial fibrillation detection in edge devices by proposing deep learning models optimized for low computational complexity. Unlike previous studies that focus primarily on accuracy, often at the expense of increased model size and computational demands, our approach achieved a balance between efficiency and performance. The proposed CNN-4 model, with only 20 289 parameters, exhibited a competitive classification performance while significantly reducing memory and power requirements, making it well-suited for real-time applications on portable devices. Additionally, the quantized version of our model achieved a 75% reduction in memory footprint while maintaining minimal losses regarding the F1-score, further reinforcing its suitability for resource-constrained environments.

Despite these promising results, certain limitations must be considered. The performance of compressed models may be affected by variations in ECG signal quality, and their generalization to different datasets requires further validation. Moreover, while our models were evaluated using a public ECG dataset, real-world deployment necessitates pre-clinical testing under diverse conditions, including different sensor types and patient demographics.

Future work will explore hybrid compression techniques, such as the combination of pruning and quantization, to further reduce model size while preserving classification accuracy. We will also investigate the feasibility of ultra-low-bit quantization (*e.g.*, fewer than eight bits) to enhance energy efficiency. Another critical direction involves deploying and optimizing our models on specialized hardware platforms, including FPGAs and System-on-Chip (SoC) architectures, in order to ensure efficient inference on edge devices.

Additionally, the development of a proof of concept for a wearable AF detection device will be pursued, integrating our models into real-time data acquisition systems. This step is crucial for validating the practical applicability of our approach in clinical settings.

Ultimately, this research contributes to advancing AF detection on edge devices, addressing the growing need for early diagnosis and intervention. By optimizing deep learning models for real-world deployment, we aim to

improve patient outcomes and support the adoption of AI-driven diagnostic tools in wearable healthcare technologies.

Acknowledgment

The authors would like to thank the CPS research group of Universidad Industrial de Santander and C-BRIC from Purdue University.

CRediT author statement

All authors: conceptualization, methodology, software, validation, formal analysis, investigation, writing (original draft, writing, review, and editing), data curation.

Conflicts of interest

The authors declare that they have no conflict of interest with regard to the publication of the results of this research.

References

[1] C. A. Morillo *et al.*, "Atrial fibrillation: The current epidemic," *J. Ger. Card.*, vol. 14, no. 3, p. 195, 2017. https://doi.org/10.11909/j.issn.1671-5411.2017.03.011.

[2] J. Primo *et al.*, "Prevalence of paroxysmal atrial fibrillation in a population assessed by continuous 24-hour monitoring," *Rev. Port. Card.*, vol. 36, no. 7-8, pp. 535–546, 2017. https://www.revportcardiol.org/en-prevalence-paroxysmal-atrial-fibrillation-in-articulo-S217420491730199X.

[3] C. Y. Chenggong Xie, Zhao Wang, J. Liu, and H. Liang, "Machine learning for detecting atrial fibrillation from ecgs: Systematic review and meta-analysis," *RCM*, vol. 25, no. 1, p. 8, 2024. https://doi.org/10.31083/j.rcm2501008.

[4] A. Y. Hannun *et al.*, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Med.*, vol. 25, no. 1, pp. 65–69, 2019. http://dx.doi.org/10.1038/s41591-018-0268-3.

[5] C. A. Fajardo, A. S. Parra, T. Castellanos-Paradaa, and K. Roy, "Low-complexity deep learning models for accurate atrial fibrillation diagnosis," in *26th Eur. Conf. Art. Intel.*, 2023. https://ecai2023.eu/conf-data/ecai2023/files/STAIRS/stairs2023_05.pdf.

[6] L. Biel, O. Pettersson, L. Philipson, and P. Wide, "Ecg analysis: a new approach in human identification," *IEEE Trans. Inst. Measure*, vol. 50, no. 3, pp. 808–812, 2001. https://doi.org/10.1109/19.930458.

[7] R. Avanzato and F. Beritelli, "Automatic ecg diagnosis using convolutional neural network," *Electronics*, vol. 9, no. 6, p. 951, 2020. https://doi.org/10.3390/electronics9060951.

[8] J. A. Castillo, Y. C. Granados, and C. A. Fajardo, "Patient-specific detection of atrial fibrillation in segments of ecg signals using deep neural networks," *Cien. Ing. Neogranadina*, vol. 30, no. 1, pp. 45–58, 2020. https://revistas.unimilitar.edu.co/index.php/rcin/article/view/4156/4249.

[9] S. Tan, G. Androz, A. Chamseddine, P. Fecteau, A. Courville, Y. Bengio, and J. P. Cohen, "Icentia11k: An unsupervised representation learning dataset for arrhythmia subtype discovery," *arXiv preprint arXiv:1910.09570*, 2019. https://arxiv.org/abs/1910.09570.

[10] CardioSTAT, "Ambulatory Cardiac Monitoring," 2021. https://www.cardiostat.com/.

[11] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," *arXiv preprint: 1707.01836*, 2017. https://arxiv.org/abs/1707.01836.

[12] Z. Xiong, M. K. Stiles, and J. Zhao, "Robust ECG signal classification for detection of atrial fibrillation using a novel neural network," *Comp. Card.*, vol. 44, pp. 1–4, 2017. https://www.cinc.org/archives/2017/pdf/066-138.pdf.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comp. Vis. Pattern Recog.*, 2016. https://doi.org/10.1109/CVPR.2016.90, pp. 770–778.

[14] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing*, vol. 520, pp. 152–170, 2023. [Online]. Available: https://doi.org/10.1016/j.neucom.2022.11.072

[15] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint: 1602.07360*, 2016. https://arxiv.org/abs/1602.07360.

[16] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Know. Disc. Data Mining*, 2006. https://dl.acm.org/doi/10.1145/1150402.1150464, pp. 535–541.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint: 1503.02531*, 2015. http://arxiv.org/abs/1503.02531.

[18] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint: 1506.02626*, pp. 1–9, 2015. http://arxiv.org/abs/1506.02626.

[19] 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," in *Proc. IEEE Conf. Comp. Vis. Pattern Recog.*, 2018. https://ieeexplore.ieee.org/document/8578384, pp. 2704–2713.

[20] SparkFun-Electronics, "Teensy 4.1. https://www.sparkfun.com/products/16771."

[21] O. A. Ademola, M. Leier, and E. Petlenkov, "Evaluation of deep neural network compression methods for edge devices using weighted score-based ranking scheme," *Sensors*, vol. 21, no. 22, p. 7529, 2021. https://www.mdpi.com/1424-8220/21/22/7529.

[22] J. Rubin, S. Parvaneh, A. Rahman, B. Conroy, and S. Babaeizadeh, "Densely connected convolutional networks for detection of atrial fibrillation from short single-lead ecg recordings," *J. Electrocard.*, vol. 51, no. 6, pp. S18–S21, 2018. https://www.sciencedirect.com/science/article/abs/pii/S0022073618303315.

[23] Q. Yao, R. Wang, X. Fan, J. Liu, and Y. Li, "Multi-class arrhythmia detection from 12-lead varied-length ECG using attention-based time-incremental convolutional neural network," *Info. Fus.*, vol. 53, no. June 2019, pp. 174–182, 2020. https://dl.acm.org/doi/abs/10.1016/j.inffus.2019.06.024.

[24] K. Fonseca, S. Osorio, J. Castillo, and C. Fajardo, "Contrastive learning for atrial fibrillation detection in challenging scenarios," in *2022 30th Eur. Signal Proc. Conf. (EUSIPCO)*. IEEE, 2022. https://doi.org/10.23919/EUSIPCO55093.2022.9909842, pp. 1218–1222.

[25] A. R. Iskandar, S. Jundi, H. Rifaı, and N. Rizoug, "Ecg classification using an optimal temporal convolutional network for remote health monitoring," *Sensors*, vol. 23, no. 3, p. 1697, 2023. https://doi.org/10.3390/s23031697.

[26] S. Asgari *et al.*, "Automatic detection of atrial fibrillation using stationary wavelet transform and support vector machine," *Comp. Biol. Med.*, vol. 60, pp. 132–142, 2015. https://doi.org/10.1016/j.compbiomed.2015.03.005.

[27] R. J. Martis *et al.*, "Automated detection of atrial fibrillation using bayesian paradigm," *Konwledge-based Syst.*, vol. 54, pp. 269–275, 2013. https://doi.org/10.1016/j.knosys.2013.09.016.

[28] M. Mohebbi and H. Ghassemian, "Detection of atrial fibrillation episodes using svm," in *2008 30th Ann. Int. Conf. IEEE Eng. Med. Biol. Soc.* IEEE, 2008. https://doi.org/10.1109/iembs.2008.4649119, pp. 177–180.

[29] S. Parvaneh, J. Rubin, S. Babaeizadeh, and M. Xu-Wilson, "Cardiac arrhythmia detection using deep learning: A review," *J. Electrocard.*, vol. 57, pp. S70–S74, 2019. https://doi.org/10.1016/j.jelectrocard.2019.08.004.

[30] A. H. Ribeiro, D. M. Tavares de Oliveira, P. R. Gomes, D. Canazart, M. Ferreira, C. R. Andersson, A. L. Ribeiro, T. B. Schon, W. Meira, and L. P. Rocha, "Automatic diagnosis of the 12-lead ecg using a deep neural network," *Nature Comm.*, vol. 11, no. 1, p. 1760, 2020.

[31] O. Yildirim, U. B. Baloglu, R. Tan, and U. R. Acharya, "Arrhythmia classification using transformer-based deep learning model," *Biomed. Signal Proc. Control*, vol. 81, p. 104477, 2023.

[32] J. Rubin, S. Parvaneh, A. Rahman, B. Conroy, and S. Babaeizadeh, "Densely connected convolutional networks for ecg classification," *J. Electrocard.*, vol. 57, pp. S20–S24, 2018. 10.1016/j.jelectrocard.2018.10.004.

[33] Y. Oh, H. Moon, D. Kim, J. Park, and D. Kim, "Transformer-based arrhythmia detection on ecg signals," *Sensors*, vol. 22, no. 8, p. 2966, 2022.

[34] I. J. Selvam, M. Madhavan, and S. K. Kumarasamy, "Detection and classification of electrocardiography using hybrid deep learning models," *Hellenic J. Card.*, vol. 81, pp. 75–84, 2025.

[35] A. R. Ismail, S. Jovanovic, N. Ramzan, and H. Rabah, "Ecg classification using an optimal temporal convolutional network for remote health monitoring," *Sensors*, vol. 23, no. 3, p. 1697, 2023. https://doi.org/10.3390/s23031697.

[36] T. Mahmud, A. Barua, D. Islam, M. S. Hossain, R. Chakma, K. Barua, M. Monju, and K. Andersson, "Ensemble deep learning approach for ECG-based cardiac disease detection: Signal and image analysis," in *2023 Int. Conf. Info. Comm. Tech. Sust. Dev. (ICICT4SD)*, 2023, pp. 70–74.

[37] Z. Zhao, X. Dong, J. Liu, Y. Guo, and Y. Yao, "ECG classification using a lightweight transformer-based model," *Biomed Signal Proc. Control*, vol. 78, p. 103931, 2022.

[38] E. Tanghatari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Federated learning by employing knowledge distillation on edge devices with limited hardware resources," *Neurocomputing*, vol. 531, pp. 87–99, 2023.