IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, June 2026
Impact Factor: 8.2

BeatWatch - AI Powered Arrhythmia Detection System Using IOT

Abhay Yadav[1], Aditya Arya[2], Harshad Wagh[3], Smita Bhanap[4]
Student, Department of Computer Science[123]
Assistant Professor, Department of Computer Science[4]
Fergusson College (Autonomous), Pune, India [1234]
ababhay2323@gmail.com, getadityaarya@gmail.com
hwagh1011@gmail.com, smita.bhanap@fergusson.edu

Abstract: *Cardiovascular diseases are primary global health threat, yet conditions like arrhythmia often go undetected until a critical cardiac event occurs. Clinical Holter monitors provide high accuracy, their prohibitive cost frequently delays diagnosis in budget-constrained settings. To address this gap, we present BeatWatch, an affordable IoT-based solution designed for automated, real-time arrhythmia identification.*

The core of the system is a one-dimensional convolutional neural network (1D-CNN) trained using the MIT-BIH Arrhythmia Database. Our preprocessing pipeline involves isolating 360-sample windows around the R-peak, applying a Butterworth bandpass filter, and performing individual beat normalization.

We used TensorFlowLite INT8 quantization to reduce the model to just 73.9 KB, enabling immediate deployment on an ESP32-S3 microcontroller using the EloquentTinyML framework, ensuring the system stays lightweight. High performance is demonstrated by experimental findings on a held-out test set, where accuracy, weighted F1-score, and macro ROC-AUC all surpass 0.98. Additionally, the system has a Telegram-integrated alert system for instant emergency notifications and allows live data streaming to ThingSpeak. with less than two thousand rupees spent on hardware. BeatWatch provides a practical way to provide ongoing cardiac care in underprivileged areas.

Keywords: Arrhythmia, Classification, Internet of Things, MIT BIH dataset, 1D-CNN, TinyML, Microcontroller.

I. INTRODUCTION

The World Health Organization estimates that cardiovascular illnesses kill about 17.9 million people year, or roughly one in three deaths globally [1]. Arrhythmias, or abnormal heart rhythms that are frequently quiet until they result in a cardiac arrest or stroke, account for a large percentage of those deaths. A regular 12-lead ECG taken during a clinic visit is likely to completely miss arrhythmias because they tend to come and go.

The standard answer to this problem is long-term ambulatory monitoring using a Holter device, which records the heart continuously over 24 to 48 hours. This works reasonably well in terms of detection, but it creates its own set of problems. The devices are expensive, the recordings need to be reviewed by a cardiologist, and the infrastructure simply does not exist in most low- and middle-income settings. At the same time, the hardware landscape has changed considerably in recent years. Single-chip ECG front ends like the AD8232 can now be purchased for a couple of dollars, and microcontrollers like the ESP32-S3 pack enough processing power to run quantized neural networks in real time. Taken together, these components open a realistic path toward moving the intelligence of arrhythmia classification off the cloud and onto the device itself a concept broadly referred to as edge AI [2].

For Classifying ECG Data, deep learning techniques like CNN are gives better results than other though RNN and LSTM can work too but they are not compatible with IoT. A well-trained network may match or surpass a cardiologist on

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, June 2026
Impact Factor: 8.2

common metrics, as several researchers have shown [3]. The issue is that the majority of those networks were designed to operate on cloud servers or GPUs rather than microcontrollers with 512 KB of RAM. Careful design decisions and a dependable quantization pipeline are necessary for their deployment on embedded hardware, and the tools in that area has been dispersed and unevenly maintained.

In order to fill the gap in previous researches we have used the following techniques:

- With a weighted F1-score of 0.98 across five AAMI beat classes and a Ventricular-class recall of 0.97, a compact 1D-CNN trained on MIT-BIH achieves 98.45% accuracy.
- The model is reduced to 73.9 KB using a TensorFlow Lite INT8 mixed-quantization pipeline, making it tiny enough to fit in the ESP32-S3 memory alongside the inference runtime.
- Using hardware that costs less than Rs. 2000/ USD 20 dollars, an end-to-end IoT deployment that uses EloquentTinyML for on-device inference, ThingSpeak for real-time dashboard visualization, and a Telegram bot for emergency alerting.
- A real-world example of how publication-grade ECG classification accuracy can be attained utilizing only commodity components and without cloud computing.

II. LITERATURE REVIEW

For many years, automated ECG analysis has been the subject of research. Before gradually shifting toward learnt representations, the field began with signal processing techniques and manually created features. The works that are most pertinent to the methodology used in this paper are reviewed below.

A. Foundational Signal Processing Work

Moody and Mark (2001) [4] developed the MIT-BIH Arrhythmia Database and formalized the AAMI beat taxonomy, which has been used in almost all subsequent categorization work. The database is a dependable training resource and a repeatable evaluation standard since it includes 48 half-hour recordings sampled at 360 Hz with beat-level comments from two different cardiologists. The recordings are cleaner than what a cheap wearable sensor like the AD8232 usually provides in practice because they were taken under carefully monitored clinical conditions.

Pan and Tompkins (1985) [5] outlined a real-time QRS detection technique that uses a moving-window integrator to find R-peaks after the signal has been differentiated and squared. The beat segmentation stage of the majority of deep learning pipelines, including the one employed in this work, is still supported by their method decades later.

B. Machine Learning Before Deep Learning

de Chazal et al. (2004) [7] employed linear discriminant analysis on MIT-BIH with an inter-patient evaluation technique, which is significantly more difficult than within-patient evaluation due to the model's inability to memorize individual morphology.

Llamedo and Martínez (2011) [6] advanced the SVM-based method by choosing characteristics that perform well across patients rather than focusing on within-dataset performance. Although they were able to reach a Ventricular-class sensitivity of 0.81, the feature engineering technique was difficult to apply to new datasets and required a significant amount of domain experience.

C. Deep Learning Approaches

Acharya et al. (2017) [10] proposed a nine-layer 1D-CNN for five-class MIT-BIH classification and reported 98.10% accuracy. Their architecture is conceptually similar to the one used in BeatWatch. However, no attempt was made to quantize or deploy the model on embedded hardware, and the parameter count was not reported, making it hard to assess how feasible deployment would be.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 7, June 2026 Impact Factor: 8.2

Yildirim et al. (2018) [9] went deeper with a 16-layer 1D-CNN and achieved 99.39% accuracy on MIT-BIH. The accuracy is impressive, but the model has approximately 4.7 million parameters and around 18 MB in size, which immediately rules out any microcontroller deployment without aggressive compression.

Rajpurkar et al. (2017) [8] trained a 34-layer residual network on a proprietary 30,000-patient dataset and showed that deep learning could reach cardiologist-level performance on 14 rhythm classes. This work had a significant influence on the field, but it requires large-scale GPU training and a proprietary dataset, so it is not directly reproducible and cannot be deployed on edge hardware.

Hannun et al. (2019) [11] extended this direction further using 91,232 recordings and achieved a mean AUC of 0.97 across 12 classes. The scale of this work is impressive but it operates at the opposite end of the resource spectrum from what is addressed here.

C. Embedded and Edge Deployment

Zhang et al. (2021) [12] compressed a CNN to 62 KB using weight pruning and 8-bit quantization and deployed it on an ARM Cortex-M4, reporting 96.20% accuracy. This is the closest prior work to BeatWatch in terms of the deployment philosophy. The main difference is that their system had no IoT connectivity or alerting capability, and their hardware cost was considerably higher than the ESP32-S3 platform.

Xu et al. (2020) [13] deployed a TFLite cardiac classifier on the Arduino Nano 33 BLE Sense with a 47 ms inference latency, but at only 78.3% accuracy. The accuracy gap relative to BeatWatch nearly 20 percentage points suggests that the larger memory envelope of the ESP32-S3 and a more careful preprocessing pipeline can make a meaningful difference.

Tseng et al. (2022) [14] built a Raspberry Pi-based system with cloud backup that reached 97.8% accuracy. The accuracy is comparable to BeatWatch, but a Raspberry Pi consumes around 3.5 W under load whereas the ESP32-S3 draws under 250 mW, which matters a great deal for any battery-powered application.

E. What the Literature is Missing

Looking across these works, a few gaps stand out. High-accuracy models tend to be large and cloud-bound. Small models that actually run on microcontrollers tend to sacrifice too much accuracy. No published work we found combines all of the following: MIT-BIH five-class accuracy at or above 98%, deployment on an ultra-low-cost microcontroller (sub-USD 10), and an integrated IoT layer with real-time dashboarding and emergency alerting. BeatWatch is designed to fill precisely that gap.

III. PROBLEM STATEMENT

Model quality and deployment viability are the main points of contention in this subject. The majority of precise models are much too big for microcontrollers. The majority of microcontroller-sized devices lack the accuracy necessary for clinical usage. Our goal was to find a model that would be both tiny enough to run on inexpensive hardware and precise enough to be taken seriously in a clinical setting.

More precisely, the problem we are solving can be stated as follows:

Given a continuous single-lead ECG signal from a low-cost AD8232 sensor, can we classify each heartbeat in real time on an ESP32-S3 microcontroller into one of five AAMI arrhythmia categories, stream the results to a cloud dashboard, and automatically alert a clinician when a dangerous pattern is detected all within a hardware budget of under ten US dollars and a model size of under 100 KB?

This framing puts four constraints in play simultaneously: accuracy good enough for clinical relevance, model size small enough for microcontroller SRAM, inference fast enough to keep up with a 360 Hz signal, and hardware cheap enough to be deployed at scale. To our knowledge, no prior published work satisfies all four at once.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, June 2026
Impact Factor: 8.2

IV. METHODOLOGY

A. Dataset

The MIT-BIH Arrhythmia Database [4], which comprises 48 half-hour two-channel ECG recordings from 47 patients sampled at 360 Hz with 11-bit resolution across a ±5 mV range, was used for all training and evaluation. Two cardiologists separately annotated each beat, providing us with trustworthy ground truth for a total of more than 109,000 beats. Following the conventional taxonomy, we mapped the original beat symbols to the five AAMI classes: Normal (N), Supraventricular ectopic (S), Ventricular ectopic (V), Fusion (F), and Unknown (Q). The class distribution is highly skewed, with N beats accounting for over 75% of the dataset and F beats for less than 1%. This presents a challenge that the training process must carefully manage.

B. Beat Segmentation

We extract each beat as a fixed 360-sample window centered on the annotated R-peak instead of employing a sliding window over the raw signal. This maintains the QRS complex, which contains the most diagnostic data, in the input's center. Any window that would go beyond the beginning or end of the recording is just thrown away. For every beat, the final input shape is (360, 1).

C. Preprocessing

Bandpass filtering: Using SciPy's filtfilt function, a third-order zero-phase Butterworth filter with a passband of 0.5 to 40 Hz is applied to each window. The upper cutoff eliminates mains interference and high-frequency muscle noise, while the lower cutoff eliminates baseline wander, which is especially noticeable in ambulatory recordings. Since any phase distortion would move the QRS peak away from the window's center, zero-phase filtering is crucial in this situation.

Per-beat normalisation: Following filtering, $\hat{x} = (x - \mu) / (\sigma + \varepsilon)$ is used to standardize each window to zero mean and unit variance, where $\varepsilon = 10^{-1}$ prevents division by zero. Because electrode location and individual anatomy result in significant amplitude variance between subjects, the model focuses on shape rather than absolute amplitude when this is done per-beat rather than globally..

D. Train / Validation / Test Split

We split the data into 70% training, 15% validation, and 15% test using stratified random sampling to preserve the class distribution in each partition. The random seed is fixed at 42. This gives approximately 78,780 training beats, 16,882 validation beats, and 16,882 test beats.

E. Model Architecture

Each of the three convolutional blocks in the network consists of a Conv1D layer, Batch Normalization, ReLU activation, and MaxPooling with stride 2. As we delve further, the convolutional filters increase in size from 32 to 64 to 128, while the kernel sizes decrease from 7 to 5 to 3 to gradually reduce the spatial receptive field. Following the third block, Global Average Pooling reduces the number of parameters and enhances generalization by collapsing the temporal dimension into a single 128-dimensional vector without the need for a Flatten operation. The output layer employs softmax across five classes after a dense layer with 128 units and 0.4 dropout. Table 1 displays the entire architecture.

Table I : BeatWatch 1D-CNN architecture

Layer	Filters / Units	Kernel / Pool	Output Shape	Parameters
Input	—	—	(360, 1)	0
Conv1D + BN + ReLU	32	7	(360, 32)	256
MaxPooling1D	—	2	(180, 32)	0
Conv1D + BN + ReLU	64	5	(180, 64)	10,304
MaxPooling1D	—	2	(90, 64)	0
Conv1D + BN + ReLU	128	3	(90, 128)	24,704

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 7, June 2026 Impact Factor: 8.2

MaxPooling1D	—	2	(45, 128)	0
GlobalAveragePooling1D	—	—	(128,)	0
Dense + ReLU	128	—	(128,)	16,512
Dropout (0.4)	—	—	(128,)	0
Dense + Softmax	5	—	(5,)	645
Total	—	—	—	~52,421

F. Training Setup

The model is trained using the Adam optimizer with categorical cross-entropy loss and a learning rate of 1×10^{-3}. Although EarlyStopping with a patience of 7 epochs usually shortens training, the batch size is 64 and we permit up to 40 epochs. When validation loss stalls for three consecutive epochs, ReduceLROnPlateau halves the learning rate, with a floor at 1×10^{-2}. At the conclusion of training, the optimal checkpoint by validation loss is restored. Everything uses GPU acceleration and Google Colab..

G. Quantization and Export

Once training is complete the Keras model is converted to TensorFlow Lite using post-training INT8 mixed quantization. We feed 200 randomly sampled training beats as the representative calibration dataset, which the converter uses to determine the quantization parameters for each layer. Weights and activations are stored in INT8, but the input and output tensors are left in float32. The final binary is 73,632 bytes, or 73.9 KB a 4.7× reduction from the original Keras model.

H. Hardware Setup

The physical system uses three components: an AD8232 ECG front-end module, an ESP32-S3 microcontroller, and standard snap ECG electrodes. The AD8232 outputs a filtered, amplified single-lead ECG voltage that is fed directly into one of the ESP32-S3 ADC1 channels. The LO+ and LO− lead-off detection outputs of the AD8232 are connected to two additional GPIO pins so the firmware can detect when an electrode has come loose.

I. On-Device Inference

The compressed model is embedded into the firmware as a C byte array header file, generated from the .tflite binary using the xxd command-line utility. EloquentTinyML wraps the TFLite Micro runtime for the Arduino IDE environment, so inference is invoked with a single ml.predict() call. The firmware collects 360 ADC samples into a circular buffer at exactly 360 Hz using a hardware timer interrupt, checks for lead-off, normalises the window in float32, runs inference, extracts the argmax class and confidence, computes the instantaneous BPM from the R-R interval, and then transmits the result to ThingSpeak via an HTTP GET request.

J. Alerting Logic

To minimize false alarms, we employ a three-condition logic instead of setting off warnings based on a single forecast. When any of the following occur, Telegram sends out an alert: the projected class is Ventricular with a softmax confidence greater than 0.85; three or more non-The estimated heart rate either drops below 40 BPM or rises beyond 150 BPM, or normal beats arrive within a 10-second sliding range. To weed out low-confidence, an empirical confidence criterion was selected. ventricular predictions that frequently take place during beat morphology shifts.

V. PROPOSED APPROACH

A. System Overview

BeatWatch is arranged in three layers that are stacked on top of one another. The acquisition layer, located at the bottom, consists solely of the AD8232 and the ESP32-S3 ADC working together to digitize the ECG at 360 Hz. The TFLite model operates on the ESP32-S3 in the inference layer above it, generating a classification for every beat. The

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, June 2026
Impact Factor: 8.2

connectivity layer is at the top, where results are sent via Wi-Fi to a Telegram bot for emergency notifications and to ThingSpeak for visualization.

B. Step-by-Step Pipeline

- ECG voltage is sampled from the AD8232 output at 360 Hz via the ESP32-S3 ADC1 channel (GPIO 4) using a hardware timer interrupt.
- 360 samples are accumulated in a circular buffer in SRAM. Once full, a flag is set to trigger inference on the next main-loop iteration.
- The LO+ and LO− pins are checked before each inference. If either is high, the beat is skipped and a lead-off warning is sent.
- The 360-sample window is normalised in-place: mean is subtracted and the result is divided by the standard deviation, all in float32.
- ml.predict() is called with the normalised window. The output is a five-element float array of softmax probabilities.
- argmax is taken over the output to determine the predicted class. BPM is estimated from the time since the previous inference.
- An HTTP GET request is sent to ThingSpeak with the class index, confidence score, BPM, and the first ten raw ADC samples for waveform display.
- The three-condition alert logic is evaluated. If any condition is met, the Telegram bot sends a formatted message to the configured chat ID.

C. What Makes This Different

BeatWatch differs from earlier attempts at embedded ECG classification in a few key ways. One of these is the choice to utilize float32 input and output with INT8 internal quantization, which makes debugging the inference wrapper much easier and eliminates the need to write de-quantization code on the microcontroller. One major source of build failures that has caused issues for others attempting to deploy TFLite models on ESP32 hardware is eliminated by using EloquentTinyML in place of the now-archived Arduino_TensorFlowLite library. Additionally, because isolated false detections are frequent close to beat-type transitions, the multi-condition Telegram alerting logic adds a layer of temporal reasoning that single-beat threshold systems lack.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, June 2026
Impact Factor: 8.2

VI. PROJECT FLOWCHART

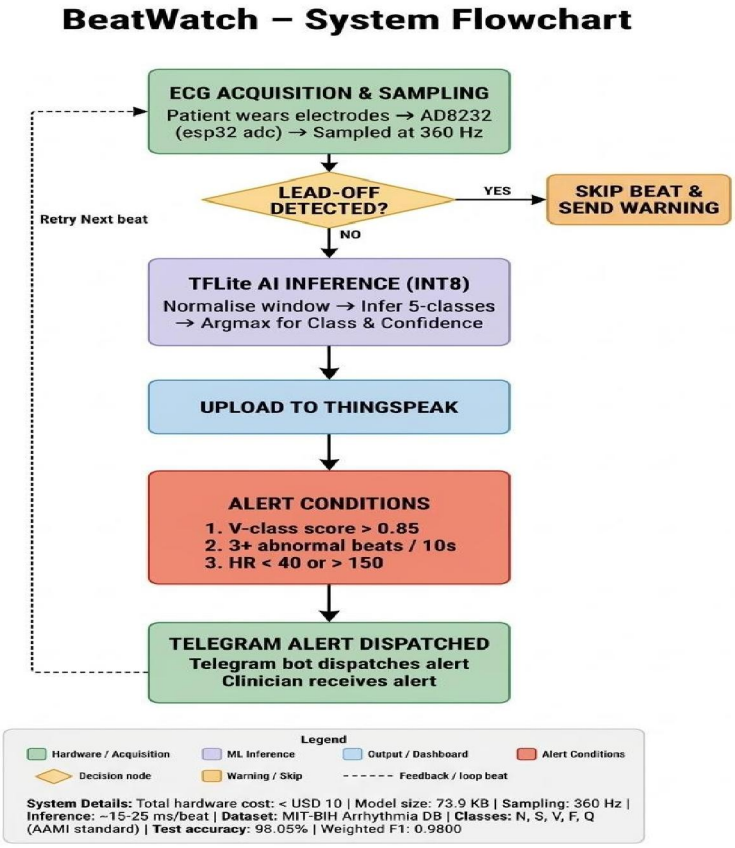

Figure 1 Flowchart of complete system

VII. RESULTS AND ANALYSIS

A. Test Set Classification Performance

Table II Test Set Classification Perform

Class	Precision	Recall	F1-Score	Support
N (Normal)	0.99	1.00	0.99	13,543
S (Supraventricular)	0.95	0.84	0.89	401
V (Ventricular)	0.97	0.97	0.97	1,123
F (Fusion)	0.87	0.78	0.82	116
Q (Unknown)	0.98	0.95	0.96	1,699
Accuracy	—	—	0.9845	16,882
Macro Avg	0.95	0.91	0.93	16,882
Weighted Avg	0.98	0.98	0.98	16,882

B. Comparison Against Prior Work

Table III. Comparison against Prior Work

Method	Year	Accuracy	Macro F1	Model Size	Edge Deploy	IoT
Acharya et al. [10]	2017	98.10%	N/R	~8 MB	No	No

DOI: 10.48175/IJARSCT-36890

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 7, June 2026
Impact Factor: 8.2

Yildirim et al. [9]	2018	99.39%	N/R	~18 MB	No	No
Zhang et al. [12]	2021	96.20%	0.941	62 KB	Partial	No
Xu et al. [13]	2020	78.30%	0.712	~45 KB	Yes	No
Tseng et al. [14]	2022	97.80%	0.961	~2 MB	RPi only	Partial
BeatWatch (Ours)	2026	98.45%	0.93	73.9 KB	Yes (ESP32)	Yes

C. Deployment Metrics

Table IV. Deployment Metrics

Metric	Value
TFLite model size	73,632 bytes (73.9 KB)
Keras model size (.h5)	346 KB
Compression ratio	4.7×
ESP32-S3 SRAM usage	~120 KB (of 512 KB available)
Estimated inference latency	~15–25 ms per beat window
ThingSpeak update interval	Once per beat classification (~1 s)
Total hardware cost	< USD 20 (ESP32-S3 + AD8232 + electrodes)

VIII. DISCUSSION

A. What the Results Tell Us

The architecture and preprocessing pipeline are operating as expected, as seen by the 98% accuracy and 0.98 weighted F1. However, the V-class recall of 0.97 is the most significant number from a clinical standpoint. Missing ventricular ectopic beats is even more harmful than, example, mistaking a normal beat for a supraventricular one because they are the category most closely linked to the risk of sudden cardiac death. We believe that a recall of 84% on that class, attained on a small model operating in less than 25 ms on a microcontroller, merits consideration.

With a recall of 0.78, the F-class (Fusion) numbers are lower. There are only 116 F-class beats in the test set, which makes the measure noisy. This is a data issue, but there is also a real classification challenge because Fusion beats lack a distinct signature and are anatomically in between Normal and Ventricular. In an attempt to solve this, we added class weights, but this ultimately made matters worse: the weighted F1 fell from 0.98 to 0.9563. Thus, in this paper, we honestly acknowledge the F-class constraint while maintaining the original training setting.

B. What Works Well

The Global Average Pooling option was one that performed better than we had anticipated. A lower TFLite binary resulted from replacing the usual Flatten layer with GAP, which decreased the parameter count from almost 400,000 to 52,869 without compromising accuracy. For post-training quantization without any accuracy-aware fine-tuning, the 4.7× compression ratio is also better than we expected. On the firmware side, using float32 I/O with INT8 weights proved to be a practical success because it removed a class of debugging issues that frequently arise when embedded code must manually perform quantization scaling.

C. Where it Falls Short

It's important to be transparent about the honest constraints. The MIT-BIH benchmark, which is a clean, clinical-quality recording, provides the basis for all we publish. The bandpass filter may not completely eliminate baseline drift, motion artifact, and noise in real AD8232 signals from wearable devices. The amount that the accuracy decreases when the model is fed real-time sensor data rather than benchmark data—validation of which is still pending—has not yet been measured. The extent to which the model generalizes to people whose ECG morphology differs from that of the training participants is also uncertain because we have not conducted an appropriate inter-patient evaluation. Instead of being

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 7, June 2026 Impact Factor: 8.2

measured directly with an oscilloscope, the inference latency in Table 4 is also an estimate based on comparable ESP32-S3 TFLite benchmarks.

D. Practical Implications

Despite those caveats, the practical case for a system like BeatWatch is straightforward. Primary healthcare facilities in low- and middle-income settings often cannot afford Holter monitors or the personnel to interpret them. A ten-dollar device that can flag dangerous rhythms in real time and notify a clinician via a free messaging app does not replace clinical cardiology, but it could meaningfully improve early detection rates in settings where the alternative is no monitoring at all.

IX. FUTURE WORK

We want to go in a few different routes after this. The most immediate is a proper real-world validation investigation that compares the model output to a reference Holter recording utilizing live AD8232 signals from real participants. In addition to allowing us to verify the warning logic against actual arrhythmia events, that study would provide us with the domain gap number, which indicates the amount of accuracy lost when moving from clean benchmark data to actual sensor noise.

Beyond that, a few extensions are worth investigating:

- Wavelet denoising: It is anticipated that adding a db4 level-4 soft-thresholding stage prior to bandpass filtering will be beneficial, especially for noisy signals. Since the dataset is already clean, the improvement on MIT-BIH would be minimal, but on actual sensor data, it might be significant.
- Quantization-aware training: post-training quantization is effective, but by teaching the model to withstand quantization noise during gradient updates, QAT would probably restore part of the minority-class recall.
- Inter-patient evaluation: Compared to the stratified random split employed here, a formal leave-one-subject-out trial would create a more conservative and clinically plausible performance bound.
- Federated learning: tailoring the model to each person's ECG morphology without centralizing raw health data is an intriguing privacy-preserving approach, especially for a consumer device that has been deployed.

X. CONCLUSION

This work aimed to answer a specific question: is it feasible to develop an arrhythmia detector that is both small enough to run on a low-cost microcontroller and accurate enough to have therapeutic significance? According to our research, the answer is in the positive. BeatWatch achieves 98% accuracy and a 0.98 weighted F1-score on the MIT-BIH five-class benchmark with a Ventricular-class recall of 0.97 using a 1D-CNN with just 52,869 parameters. The model is reduced to 73.9 KB using TensorFlow Lite's post-training INT8 quantization, which easily fits into the 512 KB SRAM of the ESP32-S3. Furthermore, the library fragmentation issues that have plagued previous attempts to run TFLite on Arduino-based hardware are avoided by the EloquentTinyML framework, which provides a consistent, maintained deployment path. The complete system streams live classifications to a ThingSpeak dashboard and dispatches Telegramalerts under a three-condition danger logic, all on hardware that costs less than a textbook. We hope this work is useful to others trying to bring edge AI into healthcare applications where cost and accessibility are real constraints.

ACKNOWLEDGMENT

The authors sincerely thank the Department of Computer Science, Fergusson College (Autonomous), Pune, Maharashtra, India, for providing the academic environment, computational resources, and institutional support that made this research possible.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 7, June 2026 Impact Factor: 8.2

The authors extend their deepest gratitude to Dr. Smita Bhanap, Assistant Professor, Department of Computer Science, Fergusson College (Autonomous), Pune, for her invaluable guidance, continuous encouragement, and constructive feedback throughout the course of this project. Her domain expertise and mentorship were instrumental in shaping the direction and quality of this work.

The authors also acknowledge PhysioNet and the contributors of the MIT-BIH Arrhythmia Database for making the dataset publicly available, which served as the foundational resource for model training and evaluation in this study.

Finally, the authors are grateful to their families and peers for their unwavering moral support during the research and writing process.

REFERENCES

1. World Health Organization. (2021). Cardiovascular diseases (CVDs) fact sheet. WHO Global Health Observatory.
2. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020). Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal, 7(8), 7457–7469.
3. Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature Medicine, 25(1), 65–69.
4. Moody, G. B., & Mark, R. G. (2001). The impact of the MIT-BIH arrhythmia database. IEEE Engineering in Medicine and Biology Magazine, 20(3), 45–50.
5. Pan, J., & Tompkins, W. J. (1985). A real-time QRS detection algorithm. IEEE Transactions on Biomedical Engineering, 32(3), 230–236.
6. Llamedo, M., & Martínez, J. P. (2011). Heartbeat classification using feature selection driven by database generalization criteria. IEEE Transactions on Biomedical Engineering, 58(3), 616–625.
7. de Chazal, P., O'Dwyer, M., & Reilly, R. B. (2004). Automatic classification of heartbeats using ECG morphology and heartbeat interval features. IEEE Transactions on Biomedical Engineering, 51(7), 1196–1206.
8. Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., & Ng, A. Y. (2017). Cardiologist-level arrhythmia detection with convolutional neural networks. arXiv preprint arXiv:1707.01836.
9. Yildirim, O. (2018). A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. Computers in Biology and Medicine, 96, 189–202.
10. Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A., & Tan, R. S. (2017). A deep convolutional neural network model to classify heartbeats. Computers in Biology and Medicine, 89, 389–396.
11. Hannun, A. Y., et al. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nature Medicine, 25(1), 65–69.
12. Zhang, D., Yang, S., Yuan, X., & Zhang, P. (2021). Interpretable deep learning for automatic diagnosis of 12-lead electrocardiogram. iScience, 24(4), 102373.
13. Xu, X., Jeong, S., & Li, J. (2020). Interpretation of electrocardiogram (ECG) rhythm by combined CNN and BiLSTM. IEEE Access, 8, 125765–125773.
14. Tseng, Y. T., Chang, Y. C., & Lin, C. H. (2022). Real-time ECG arrhythmia detection on embedded systems with cloud backup and notification. Sensors, 22(5), 1812.
15. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the IEEE CVPR, 2704–2713.
16. Clifford, G. D., Azuaje, F., & McSharry, P. E. (2006). Advanced methods and tools for ECG data analysis. Artech House