Embedded DNN Classifier for Five Different Cardiac Diseases

Muhammad Shakeel Akram, Bogaraju Sharatchandra Varma, Dewar Finlay
School of Engineering
Ulster University
Belfast, United Kingdom
Email: {akram-ms, s.bogaraju, d.finlay}@ulster.ac.uk

***Abstract*—The evolution of modern healthcare has been significantly shaped by the convergence of connected sensors, smart Wearable Devices, Artificial Intelligence, and the Internet of Things giving rise to the domain of eHealth and offering invaluable insights into the complications of heart health. eHealth's impact extends to facilitating diagnosis, treatment, and medication for a diverse array of conditions, prominently including cardiac diseases. Despite substantial strides in medical technology, the detection of arrhythmia remains a persistent challenge, with early diagnosis holding the potential to avert numerous fatalities. This paper proposes an ultra-lightweight (876KB) Embedded-Deep Neural Network model specifically designed for resource-constrained devices. With high accuracy ranging from 94% to 99% for five classes identified from the MIT-BIH dataset, the proposed model is small enough to fit on tiny devices like the Arduino Nano BLE 33 Sense. This translates to low power consumption and real-time inference, making it ideal for screening cardiac diseases on wearable devices.**

I. Introduction

Globally, 50 million individuals face a significant risk from Cardiac diseases (CDs) [1]. These CDs account for nearly 37% of worldwide deaths. The threat of CD-related fatalities is projected to rise, given the anticipated growth of the elderly population to 1.4 billion by 2030 and 2.1 billion by 2050 [2], [3]. Among CDs, arrhythmia is prevalent and affects 1 out of 1000 people annually [4]. Among Life-Threatening CDs (LTCDs), ventricular arrhythmias account for 80% of LTCDs and are hard to diagnose, often leading to rapid mortality (annually over 350K in the USA) within minutes [5], [6], necessitating prompt diagnosis and treatment to prevent these damage.

Cardiac diseases encompass electrical (arrhythmia), circulatory (blood vessels disorder), and structural (heart muscle diseases) issues [1]. Various parameters contribute to arrhythmia diagnosis, including heart rate from ECG and PPG, EEG, MRI, CT scan, heart rate variability, stroke volume, cardiac output, systolic time intervals, left ventricular ejection time, pre-ejection period, systolic time ratio from Impedance Cardiography (ICG), chaotic electrical signals, and clinical data. Indicators like chronic obstructive pulmonary disease, renal dysfunction, vascular disease, and valvular disease are also significant in identifying CDs. ECG signals stand out as highly effective for real-time independent systems in CDs diagnosis [7].

979-8-3503-5298-6/24/$31.00 ©2024 IEEE

Considerable research and development endeavours have focused on creating machine learning solutions using Convolution Neural Networks (CNNs) and Deep Neural Networks (DNNs) to analyze vital ECG data for prompt CDs diagnosis [8]–[10]. Some models specialize in specific arrhythmias, while others focus on normal and abnormal rhythm classifications. Additionally, some larger models demand substantial memory space, flash memory, and computational capabilities, highlighting a gap for improvement and an opportunity to leverage the abundance of available embedded devices in healthcare. The need arises for automatic, cost-effective, low-power, real-time, and efficient AI-based detection of CDs, which holds the potential to revolutionize *low-power device diagnosis*.

We propose an Embedded DNN classifier for five different CDs using TinyML [11]. This model is trained on engineered data and implemented on an Arduino BLE Sense board. The model achieves very high accuracies ranging from 94% to 99% for each class with a compact size of 876KB. The model achieves a very good F1 score considering the utilization of a large dataset comprising 305,066 pre-processed samples segmented from the MIT-BIH dataset. We discuss 24 distinct implementations enabling us to study the characteristics of the model to come up with a highly accurate tiny model. This compact Embedded-DNN classifier can be readily adapted for resource-constrained wearable devices, enabling real-time classification of CDs. This pursuit aligns with the larger goal of enhancing diagnostic capabilities while utilizing the advantages of diverse embedded devices [12].

The organization of the paper is as follows. Background is discussed in section II. Details about the processes involved in model development and dataset preparation are elaborated in Section III. Subsequently, Section IV provides a comprehensive overview of the achieved results, accompanied by details on various implementations suitable for different resource-constrained embedded devices. Related work with a comparison to our work is presented in Section V followed by the conclusion in Section VI.

II. Background

AI algorithms play a pivotal role in learning complex patterns from data, showcasing significant potential in CD diagnoses reliant on digitized patient-specific information like

ECGs. Three main types of algorithms are prevalent: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning holds particular importance in many CD diagnosis methods, including SVM, KNN, GNNs, LSTM, CNN, and DNN. CNNs and DNNs have emerged as effective tools for processing ECG signals [13], [14]. CNN leverages local correlation and translation invariance similar to some image signals [13], while DNNs excel in recognizing patterns and extracting valuable features from raw input data without extensive preprocessing, feature engineering, or handcrafted rules. This makes them particularly suitable for interpreting ECG data.

TinyML represents a research direction aimed at enabling ML processing on embedded systems [11]. This is achieved through the implementation of bare-metal and lightweight inference libraries, such as TFLite Micro, TVM, CMSIS-NN, and TinyEngine, coupled with the removal of unnecessary functionalities, including debugging. Additionally, model weights and computational graphs can be simplified to minimize memory and computational overhead. Edge processing also presents a significant challenge in shrinking the computational requirements of advanced ML-based near-sensor data analysis algorithms. These challenges are particularly pronounced within the mW-range of always-on battery-powered systems with processing capabilities exceeding 1GOPS, operating on small battery power and energy budgets, computation ranging from 1MHz to 400MHz, 2KB to 4MB Flash, and 32KB to 2MB storage [11], [15]. Therefore model must fit within the device memory, ensuring compatibility with embedded devices constrained by limited memory, power, and computational capacity while capturing the complexity of the data for effective computation. On-device ML capabilities in embedded systems offer a solution to these challenges, addressing concerns and potentially improving misdiagnosis rates [16]–[23].

III. Experimental Setup

The overall development process is depicted in the block diagram shown in Fig.1 and the Algorithm 1. The Embedded-DNN is implemented and tested on the Arduino BLE 33 Sense [24], utilizing the following tools: Arduino IDE [25] for embedded design, TensorFlow [26] for DNN development, TinyML-Gen [27] for C code generation for microcontrollers from TensorFlow trained models, TensorFlow Lite Micro (TFlite-micro) [28] for running ML models on microcontrollers. Physionet's MIT-BIH Arrhythmia Dataset [29], [30] is used for developing the model.

The pre-processed dataset was organized into 1x187 full precision floating point (Float32) samples, distributed equally across five categories: start=0

1) Normal beat (N/Class 0)
2) Supra-ventricular premature beat (S/Class 1)
3) Ventricular escaped beat (E/Class 2)
4) Fusion of ventricular and normal beat (F/Class 3)
5) Unclassifiable beats (Q/Class 4)

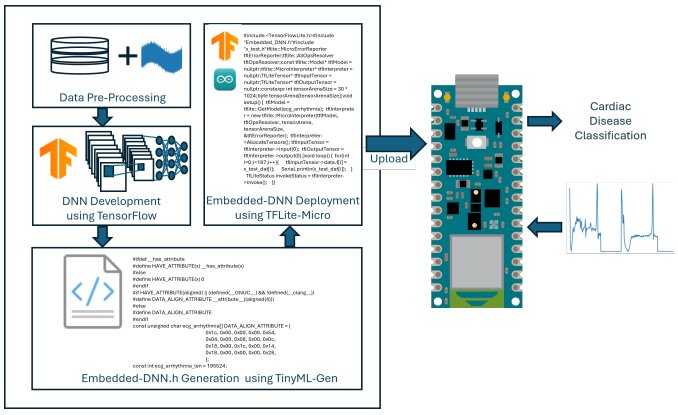

Fig. 1. Block Diagram explaining Embedded-DNN Deployment Processes.

Algorithm 1: Training and Deployment of Embedded-DNN on Arduino BLE 33 Sense

Data: Trained DNN model in TensorFlow
Result: Arduino sketch for inference of 5 Cardiac Diseases

Embedded-DNN Training using TensorFlow:;
begin
- Embedded-DNN model using TensorFlow Keras layers;
- Train the model with found optimization parameters;

Convert Embedded-DNN Model using `tinyml-gen`:;
begin
- Run `tinyml-gen` to convert the TensorFlow model into C code i.e. Embedded-DNN.h;
- Include the generated files i.e. Embedded-DNN.h in the Arduino sketch;

Arduino Sketch for Embedded-DNN Inference using tflite-micro:;
begin
- Include necessary `tflite-micro` library files;
- Initialize the TensorFlow Lite interpreter;
- Implement functions for model initialization and inference;

Compile and Upload to Arduino BLE 33 Sense:;
begin
- Compile the Arduino sketch and upload it to the Arduino board;

Run Inference on Arduino:;
begin
- Power up the Arduino and monitor the output for inference results;

In total, the dataset comprises 305,066 samples, with a distribution of 250,000 (85%) for training, 44,120 (15%) for validation, and 10,946 for testing. The testing samples were never exposed to the model during training. In addition to this, noise present in the ECGs of some of the CDs makes it difficult to diagnose and needs to be addressed. Therefore, Gaussian noise was added to the dataset involved in the training process i.e. training and validation to enhance the model's efficiency, with a mean value of 0 and a standard deviation of 0.05.

The Embedded-DNN is built using TensorFlow and comprises a total of 10 layers with the same padding, encompassing an input layer with one input channel and output layers featuring five output channels. Among these, eight hidden layers are composed of both convolutional and dense layers. The entire model encompasses 35,296 trainable parameters. The optimization process employs the Adam optimizer, while the loss function is based on categorical cross-entropy, learning rate of 0.001, ReLU activation, and Top-1 accuracy is observed.

The trained model was converted using the TinyML-Gen into a form that could be deployed on the Arduino Nano 33 BLE Sense, which is a small but powerful board-based SoC with an Arm Cortex-M4F 32-bit processor running at 64 MHz. The board has 1 MB of flash memory and 256 kilobytes (KB) of RAM. To ensure that the model could fit within the available memory and provide accurate inferences, it was developed with the memory size of the board in mind.

Finally, the converted model for the inference was deployed on Arduino Nano 33 BLE Sense using Arduino IDE and TFlite-micro. We developed a controller capable of efficiently classifying five different cardiac diseases on the device.

IV. Results and Discussion

The model fitting in the Arduino flash and performing inference with high accuracies has been achieved by exploring various pre-processing steps and training methods. In this paper we discuss twenty-four relevant experiments having different combinations of dataset pre-processing and split for training, testing and validation, different combinations of DNN layers and different combinations for each layer's neurons and kernels. Four different settings are used for preparing the training and validations data:

1) 21500 unbalanced Validation samples and 391 Unbalanced Test Samples are used for experiment setup number 1-3, 6-18, and 21.
2) 10946 unbalanced Validation samples and 10946 unbalance Test samples are used for experiment number 4.
3) Balanced validation samples were created by resampling. 44120 of these and unbalanced 10946 Test samples are used for experiment number 5, 19, 20, 22, and 24.
4) 44120 resampled and balanced samples are used for Validation and for Test. The resampled values are divided equally into Validation and Test datasets for experiment setup number 23.

To enhance model training, specific configurations were used to adjust the batch size and epochs. Training with batch size of 32 and 8 epochs was performed experiments 1-11, 13-19, 21, and 23. Experiments 20, 22, and 24 were conducted with batch size 32 and 15 epochs. For experiment number 12, we employed a batch size of 100 and 10 epochs. These tailored settings were crucial in optimizing the training process and achieving meaningful results. Each experiment set's findings provided the base for the next experiment to reach the highly accurate model within the available limits.

TABLE I
Model size and inference errors for the experiments

Exp.#	***Trainable Parameters***	***TinyML-Gen Model Size***
1	389701	9405KB Does not fit BLE-FLASH
2	387781	9356KB Does not fit BLE-FLASH
3-5	328389	7936KB Does not fit BLE-FLASH
6	195077	4714KB Does not fit BLE-FLASH
7	183033	4424KB Does not fit BLE-FLASH
8	125765	3028KB Does not fit BLE-FLASH
9	98725	2394KB Inference Error: Arena size is too small for all buffers. Needed 71936 but only 29280 was available.
10	48413	1206KB Inference Error: Arena size is too small for all buffers. Needed 35968 but only 29536 was available.
11	46331	1132KB Inference Error: Arena size is too small for all buffers. Needed 33928 but only 29536 was available.
12	48389	1179KB Inference Error: Arena size is too small for all buffers.
13	44790	1094KB Inference Error: Arena size is too small for all buffers. Needed 32608 but only 29536 was available.
14	43249	1057KB Inference Error: Arena size is too small for all buffers. Needed 31472 but only 29568 was available.
15	41708	1020KB Inference Error: Arena size is too small for all buffers. Needed 30368 but only 29568 was available.
16	40167	983KB Fits BLE-FLASH and performs Inference.
17	37085	909KB Fits BLE-FLASH and performs Inference.
18	35544	871KB Fits BLE-FLASH and performs Inference.
19,20	35296	876KB Fits BLE-FLASH and performs Inference.
21-23	32462	797KB Fits BLE-FLASH and performs Inference.
24	22046	547KB Fits BLE-FLASH and performs Inference.

Table I summarizes the model trainable parameters and size of the model deployable on BLE sense board. Table I and Figure. 2 are distributed in three major sections based on Flash and RAM available on Arduino BLE 33 Sense.

1) Experiments Number 1-8 encounter a challenge as they exceed the Flash capacity and cannot be deployed on the selected device for this particular experiment. However, Figure 2 clearly illustrates that the initial six experiments yield high overall accuracy as well as accuracy for each class. Despite their final model size ranging between

9405 KB to 4714 KB, surpassing the BLE-FLASH limit, these models can be successfully deployed on various other embedded devices with larger Flash memory. Examples of such devices include Arduino Nano RP2040 Connect, Raspberry Pi Zero 2 W, Raspberry Pi Pico W, ESP32-WROOM-32, and others.

2) Experiments Number 9-16 represent the section facing the challenge of inference errors due to an inadequate arena size for all buffers when deployed on the selected embedded device for this specific experiment. Similar to the initial set of experiments, this section demonstrates higher accuracy metrics particularly for each class in experiments 9 and 13-15. These models can be effectively deployed on BLE but they encounter inference failures attributed to insufficient arena size, preventing the coverage of all buffers during inference. Despite this challenge for deployment on Arduino BLE 33 Sense, these models can be successfully deployed on various embedded devices with equal or greater flash memory and higher RAM compared to Arduino BLE 33 Sense. Suitable devices encompass Arduino Nano RP2040 Connect, Raspberry Pi Zero 2 W, Raspberry Pi Pico W, ESP32-WROOM-32, and others.
3) Finally, Experiments 17-24 cover the section of successfully deploying the Embedded-DNN and conducting the inference on Arduino BLE 33 Sense. This section highlights superior performance in experiments 17-21 when tested on the device. Beyond exclusively showcasing these models for Arduino BLE 33 Sense, the models in this section opens the door for embedded deployment on any device with flash memory exceeding 547 KB.

In summary, all these experiments consistently yield accuracies surpassing 90%, showcasing notable and acceptable results for the classification of five different cardiac diseases. Despite some outliers due to insufficient device memory, the models exhibit a size range from 797 KB to 9405 KB, underscoring the versatility of the Embedded-DNN model. This adaptability significantly broadens the range of devices where the model can be effectively deployed.

The experiments done yield significant insights into the training process. Particularly, training on an unbalanced validation dataset results in bias, as evidenced by Experiment numbers 1-3, 6-18, and 21. Moreover, it is evident that a higher batch size does not necessarily translate to a more accurate model, as observed in Experiment number 12. The distribution of the resampled test dataset between validation and test does not accurately represent true accuracy, as indicated by Experiment number 23. Particular experiments, such as numbers 7 and 16 (UnBalanced Validation and fewer test samples, batch size 32, and 8 epochs), 12 (UnBalanced Validation and less test samples, batch size 100, 10 epochs), and 23 (distribution of resampled dataset between validation and test, batch size 32, and 8 epochs), showcase that the worst results are achieved under certain conditions. Additionally, it is highlighted that a higher number of epochs, below which model overfitting occurs, can contribute to improvements in model performance, as demonstrated by Experiment numbers 19 and 20.Effective Embedded-DNN model training requires thoughtful considerations in dataset balancing, batch size selection, and epoch tuning.

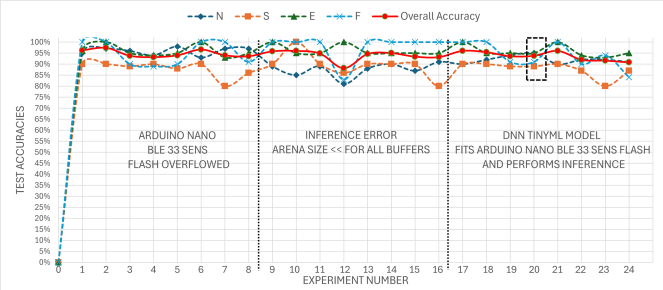

Fig. 2. Each Cardiac Disease's achieved accuracy as well as an overall accuracy chart against each experiment setting.

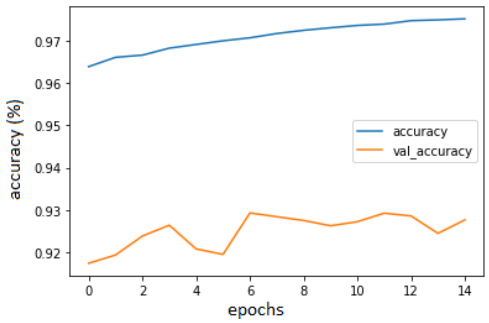

Fig. 3. Training vs validation Accuracy.

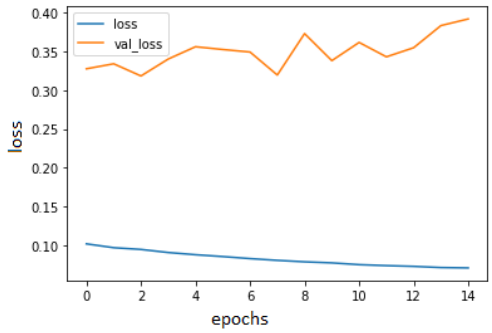

Fig. 4. Training vs validation Loss.

The training and validation outcomes of the Embedded-DNN in experiment number 20 are clearly illustrated in Figure 3 and 4. During training, the accuracy reached 97.51%, while the validation accuracy attained 92.77%. This progress was achieved within a training time of 1 hour 35 minutes 32 seconds of CPU time, and 58 minutes 22 seconds of wall time. The decision to select experiment number 20 for deploying the final highest-performing model on Arduino BLE 33 Sense is based on the following considerations. Despite experiments 17, 18, and 21 displaying higher performance in Figure 2, it's

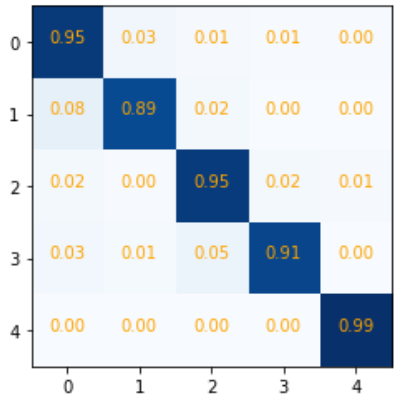

Fig. 5. Model's Confusion Matrix.

noteworthy that they were trained on unbalanced validation samples. This led to bias in the model towards certain classes, evident in the graph showcasing distinct accuracy variations between classes. Additionally, their testing was conducted on a very small set of only 391 samples, raising concerns about the representativeness of the test set in capturing the broader data distribution. Experiment 19, although providing nearly identical results to the selected experiment 20, holds preference. Both were trained on balanced training and validation datasets to mitigate bias. However, experiment 20 stands out as it was trained with a batch size of 32 and 15 training epochs, achieving a slightly higher test accuracy of 95% compared to 94% for N/Class 0. Figure.5 represents the confusion matrix on test datasets for each class. The F1 score was also better, measuring 0.82604 in comparison to 0.79127.

V. Related Works

The existing works related to classifying more than just normal and abnormal rhythms are detailed in Table II. This table provides insights into the methods, datasets, achieved performance, and limitations of each work, offering a comparison of our contributions. The table includes references (Ref.), Accuracy (A), Sensitivity (S), Specificity (Sp), and Precision (P) as key evaluation criteria. These metrics, such as Accuracy, Positive Predictive Value (PPV), F1 score, and Model Size, collectively offer a comprehensive understanding of performance, efficiency, and resource allocation.

The proposed Embedded-DNN model emerges as an accurate classifier of five cardiac diseases (CDs) on an embedded device having a remarkably small model size of 876 KB. Compared with [2], despite a slight accuracy drop of 4.85% and fewer classes, our model is trained with a larger set of 305,066 samples and supports inference on resource-constrained devices. In contrast to [31], our model classifies five classes instead of six. However, our model exhibits a higher accuracy by 0.82% and also facilitates inference on resource-constrained embedded devices. Regarding [8], our model, while experiencing a modest accuracy drop of 3.32% and handling fewer classes, surpasses by using a larger sample size of 305,066 and enabling inference on resource-constrained devices.

Similarly, compared to [9], our model demonstrates a slight accuracy reduction of 4.10%, focusing on five classes with a larger sample size of 305,066, ensuring diversity and enabling inference on resource-constrained devices. Likewise, in relation to [10] and [32], our model maintains competitive accuracy with only a 3.5% and 3% drop, respectively. However, it uses a larger sample size of 305,066, ensuring diversity and supporting inference on resource-constrained devices. Importantly, our model extends classification to five classes, outperforming the related work that focuses on four classes.

TABLE II
Embedded-DNN Comparison with State-of-the-art Relevant Solutions

Methods	***Performance***	***Limitations***
SVM Classifier for 17 Classes, GA feature selection and parameter optimization, MIT-BIH [2]	A: 98.85% S: 90.20%, Sp: 99.39%	Limited (512) ECGs and not an embedded device implantation.
CNN Classifier for 6 Classes, Daubechies WT, MIT-BIH, and CUDB [31]	A: 93.18%, S: 95.32%, Sp: 91.04%, PPV: 91.41%	Low Accuracy and not an embedded device implantation.
U-net DNN Classifier for 5 Classes, auto-encoder, and MIT-BIH [8]	A: 97.32%	Limited (47 patient) data and and not an embedded device implantation.
CNN-LSTM classifier for 6 Classes, and MIT-BIH [9]	A: 98.10%, S: 97.50%, Sp: 98.70%, PPV: 98.69%	Limited (1004 ECGs) data and not an embedded device implantation.
DNN Classifier for 5 Classes, Modified Frequency Slice WT, and MIT-BIH [10]	A: 97.5%, S:71.4%, Sp: 67.2%	Limited (47 patient) data and not an embedded device implantation.
Robust Deep Dictionary Learning Classifier for 4 Classes, Hand-crafted Input Features, and MIT-BIH [32]	A: 97.0%, S: 16.9%, Sp: 67.2%	Low sensitivity and Specificity. Classifying only 4 classes.
DNN Classifier for 5 Classes, and MIT-BIH **[Our]**	**A: 94%, F1: 0.826, Model Size: 876KB, Embedded, 305,066 data samples**	–

VI. Conclusion

Cardiac Diseases (CDs) have a significant impact on a diverse population, with a particularly pronounced effect on the elderly. The ageing demographic is projected to grow further, placing additional strain on healthcare systems, despite substantial progress in eHealth. CDs pose life-threatening risks due to their time-sensitive nature, necessitating rapid diagnosis and treatment that could be facilitated by Real-Time Independent Systems (RTIS).

While extensive research has explored various Machine Learning (ML) solutions for CDs diagnosis and treatment, Convolutional Neural Networks/Deep Neural Networks

(CNN/DNN) have demonstrated superior performance. However, the imperative lies in optimizing these ML algorithms for reduced power and time consumption while preserving performance metrics. Addressing this challenge involves compressing models, and implementing real-time AI on the edge or embedded devices, encompassing communication efficiency, computation efficiency, heterogeneity, the significance of wearable devices and their role in remote health support.

This paper provides a solution to implement a CD classifier on embedded devices with very good accuracy. The model is deployed on Arduino Nano BLE 33 Sense for the efficient classification of the five different CDs with 94% accuracy while having only 876KB of model size. This makes it extremely useful for real-time diagnosis of CDs on resource-constrained devices without communicating with the server. This paper also provides an exploration framework wherein twenty-four different models were developed and carefully tweaked to come up with an optimum design that can fit in a tiny embedded device while still maintaining accuracy. The findings underscore the importance of thoughtful considerations in model parameters, dataset balancing, batch size selection, and epoch tuning for effective Embedded-DNN model training. The insight from this paper can easily be adapted to other healthcare domains, fostering the development of real-time independent systems playing an important role in improving the quality of care and enhancing the well-being of patients worldwide.

REFERENCES

[1] H. R. Society, "Heart rhythm disorders," 2022. Available at: https://upbeat.org/heart-rhythm-disorders.

[2] P. Pławiak, "Novel methodology of cardiac health recognition based on ecg signals and evolutionary-neural system," *Expert Systems with Applications*, vol. 92, pp. 334–349, 2018.

[3] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ecg segments with convolutional neural network," *Information sciences*, vol. 405, pp. 81–90, 2017.

[4] J. Brugada, "Cardiac arrhythmias and sudden death," *e-journal of the ESC Council for Cardiology Practice*, vol. 2, January 2004.

[5] S. Sahoo, B. Kanungo, S. Behera, and S. Sabut, "Multiresolution wavelet transform based feature extraction and ecg classification to detect cardiac abnormalities," *Measurement*, vol. 108, pp. 55–66, 2017.

[6] E. J. Benjamin, M. J. Blaha, S. E. Chiuve, M. Cushman, S. R. Das, R. Deo, S. D. De Ferranti, J. Floyd, M. Fornage, C. Gillespie, *et al.*, "Heart disease and stroke statistics—2017 update: a report from the american heart association," *circulation*, vol. 135, no. 10, pp. e146–e603, 2017.

[7] C. W. Tsao, A. W. Aday, Z. I. Almarzooq, A. Alonso, A. Z. Beaton, M. S. Bittencourt, A. K. Boehme, A. E. Buxton, A. P. Carson, Y. Commodore-Mensah, *et al.*, "Heart disease and stroke statistics—2022 update: A report from the american heart association," *Circulation*, vol. 145, no. 8, pp. e153–e639, 2022.

[8] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, "Automated beat-wise arrhythmia diagnosis using modified u-net on extended electrocardiographic recordings with heterogeneous arrhythmia types," *Computers in biology and medicine*, vol. 105, pp. 92–101, 2019.

[9] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya, "Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats," *Computers in biology and medicine*, vol. 102, pp. 278–287, 2018.

[10] K. Luo, J. Li, Z. Wang, and A. Cuschieri, "Patient-specific deep architectural model for ecg classification," *Journal of healthcare engineering*, vol. 2017, 2017.

[11] V. J. REDDI, "Fundamentals of tinyml," 2022. Available at: https://learning.edx.org/course/course-v1:HarvardX+TinyML1+3T2020/home.

[12] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–8, IEEE, 2017.

[13] Y. Wei, J. Zhou, Y. Wang, Y. Liu, Q. Liu, J. Luo, C. Wang, F. Ren, and L. Huang, "A review of algorithm & hardware design for ai-based biomedical applications," *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 2, pp. 145–163, 2020.

[14] G. Quer, R. Arnaout, M. Henne, and R. Arnaout, "Machine learning and the future of cardiovascular care: Jacc state-of-the-art review," *Journal of the American College of Cardiology*, vol. 77, no. 3, pp. 300–313, 2021.

[15] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "Gap-8: A risc-v soc for ai at the edge of the iot," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 1–4, IEEE, 2018.

[16] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature medicine*, vol. 25, no. 1, pp. 65–69, 2019.

[17] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

[18] M. S. Abdelfattah, Ł. Dudziak, T. Chau, R. Lee, H. Kim, and N. D. Lane, "Best of both worlds: Automl codesign of a cnn and its hardware accelerator," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[19] J. Tang, D. Sun, S. Liu, and J.-L. Gaudiot, "Enabling deep learning on iot devices," *Computer*, vol. 50, no. 10, pp. 92–96, 2017.

[20] G. Klas, "Edge computing and the role of cellular networks," *Computer*, vol. 50, no. 10, pp. 40–49, 2017.

[21] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *computer*, vol. 50, no. 10, pp. 58–67, 2017.

[22] P. B. Soundarabai, P. Augustine, and S. Vinod, "Demystifying the edge ai paradigm," *Applied Edge AI: Concepts, Platforms, and Industry Use Cases*, p. 23, 2022.

[23] D. Sopic, A. Aminifar, A. Aminifar, and D. Atienza, "Real-time event-driven classification technique for early detection and prevention of myocardial infarction on wearable systems," *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 5, pp. 982–992, 2018.

[24] Arduino, "Nano 33 ble sense," 2024. Accessed 2024-02-06: https://docs.arduino.cc/hardware/nano-33-ble-sense/.

[25] Arduino, "Arduino ide," 2023. Accessed 2024-02-06: https://www.arduino.cc/en/software.

[26] TensorFlow, "Tensorflow," 2024. Accessed 2024-02-06: https://www.tensorflow.org/.

[27] S. Salerno, "Generate c code for microcontrollers from tensorflow models," 2020. Accessed 2024-02-06: https://pypi.org/project/tinymlgen/.

[28] TensorFlow, "Tensorflow lite for microcontrollers," 2024. Accessed 2024-02-06: https://github.com/tensorflow/tflite-micro.

[29] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, ..., and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation [Online]*, vol. 101, no. 23, pp. e215–e220, 2000.

[30] A. C. Community, "Arrhythmia prediction on ecg data using cnn." Online:Cainvas AI-Tech Systems platform, April, 2022.

[31] U. R. Acharya, H. Fujita, S. L. Oh, U. Raghavendra, J. H. Tan, M. Adam, A. Gertych, and Y. Hagiwara, "Automated identification of shockable and non-shockable life-threatening ventricular arrhythmias using convolutional neural network," *Future Generation Computer Systems*, vol. 79, pp. 952–959, 2018.

[32] A. Majumdar and R. Ward, "Robust greedy deep dictionary learning for ecg arrhythmia classification," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 4400–4407, IEEE, 2017.