Edge2Analysis: A Novel AIoT Platform for Atrial Fibrillation Recognition and Detection

Jiarong Chen, Yingfang Zheng, Yingshan Liang, Zehui Zhan, Mingzhe Jiang, Daniel S. da Silva, Wanqing Wu, and Victor Hugo C. de Albuquerque, Senior Member, IEEE

Abstract—Atrial fibrillation (AF) is a common arrhythmia disorder associated with high mortality and morbidity. When atrial fibrillation occurs, the blood supply efficiency and blood transmutation function are seriously affected. Moreover, AF can increase the risk of heart failure and ischemic stroke [1]. Approximately one-third of strokes are related to atrial fibrillation [2], [3]. According to the World Health Organization, approximately 90 million people have atrial fibrillation, accounting for 1 to 2 percent of the world’s population [4], [5]. For example, more than 7.9 million people are suffering from atrial fibrillation in China, and the atrial fibrillation incidence is related to age [6]. Therefore, atrial fibrillation urgently needs the attention of the medical industry. There are three kinds of atrial fibrillation: paroxysmal AF, persistent AF, and permanent AF [7]. Unfortunately, some atrial fibrillation patients may have palpitations, chest tightness and panic attack symptoms, while others do not have any obvious symptoms. However, for subjective unawareness, many patients with atrial fibrillation do not receive timely treatment until it progressively worsens, and the consequence of delayed treatment can cause paroxysmal AF that can evolve into permanent AF and lead to serious complication [1], such as atrial fibrillation with heart failure and atrial fibrillation with stroke. Therefore, AF detection, especially in the asymptomatic stage, is of great importance and necessity in AF treatment.

Electrocardiogram (ECG) can depict the differences between normal and atrial fibrillation signals in both atrial activity that represented by P-waves and f-waves and ventricular activity that represented by QRS complex, play a significant role in AF diagnosis. The special ECG recordings of AF patients are characterized by irregular RR intervals variability and the P-wave absence. ECG interpretation poses a great challenge for medical practices, with approximately 3 million electrocardiograms being recorded each year worldwide. Since manual interpretation is tedious and time-consuming, artificial intelligence (AI) technology has been utilized to identify AF from massive ECG data automatically. Machine learning (ML) is one of the distinctive artificial intelligence approaches and was proved to perform similarly to or even better than cardiologists [11]. Recent studies yielded some encouraging results, encourage people to use these techniques [12]. Convolutional neural networks (CNN) with multiple layers or densely connected structures were used by many researchers [2], [13]–[16]. A 94-layer residual neural network was proposed to classify different ECG rhythms [16]. Combined with recurrent neural networks (RNN), Limam [17] completed classification based on a convolutional recurrent neural network (CRNN), and similar hybrid neural network architectures, like CNN combined with Long short-term memory (LSTM), were proposed in [18], [19] to improve model performance. The attention mechanism is used to
We select suitable features to complete an AF recognition task. Our model is lightweight and has strong anti-noise ability, security [41]. On the other hand, edge computing requires advantages, such as high response speed [40], low cost and data privacy. On the one hand, edge computing offers computational complexity, and response delay. The drawbacks are prominent in places lacking adequate computing and online resources, such as underdeveloped countries, remote rural areas, and some retirement communities. In addition, it is worth noting that the health situation is bleak in these places due to poverty, harsh living conditions, and scarce healthcare resources. Therefore, researchers [26] suggested a platform where doctors can monitor patients remotely, which may promote personalized healthcare [27] and become a useful approach in telemedicine. More cost-effective and feasible methods of AF detection are appealing. Some researchers have proposed combining ECG automated analysis with wearable devices [1] or smartphones [28]. In this paper, wearables and smartphones are called edge devices, which are able to conveniently collect and process physiological signals in real-time [29]. For example, the literature [1] developed a photoplethysmography (PPG)-based AF detection on a smartwatch, which can be a feasible solution for AF real-time and longitudinal monitoring. Unlike PPG, ECG directly measures the hearts physiological activity, which has an advantage over PPG in monitoring atrial fibrillation. Another research [30] mentioned this idea, which deployed the trained ECG classification decision tree inside the edge node to measure peoples health status in home scenarios, yet with the downside that it was only an offline model. Regarding the problem of limited resources in edge devices, some researchers have made several attempts, like the quantized neural network [31], pruning model structure [32] to reduce the complexity, and a novel and effective compression tool to implement the lightweight model in microcontroller platform [33]. In addition, a binarized convolutional neural network [34] has been proposed to deploy ECG automatic classifier on wearable devices. All of these methods have the advantage of low power consumption. Among the studies, few are concerned with the performance of embedded devices, and even fewer works are similar to this research, which uses a general embedded device to achieve atrial fibrillation monitoring tasks.

Many studies combine the Internet of Things with artificial intelligence, which is called Artificial Intelligence of Things (AIoT) [30], [35]–[37]. For example, in the case above, people can check their health status on a mobile phone/device without going to the hospital to find a doctor for evaluation. Edge computing is an important concept and method to realize AIoT. Edge computing is pushing applications and services geographically closer to where these services are requested [38], [39]. This type of computing has the potential to solve the shortcomings of cloud computing. On the one hand, edge computing offers advantages, such as high response speed [40], low cost and data security [41]. On the other hand, edge computing requires a large amount of computing resources [42], which is difficult for resource-constrained [43] embedded devices to implement. Many researchers have studied the implementation of edge computing. For example, [44] built a tiny neural network for environmental predictions with acceptable memory consumption and accurate prediction. Some researchers [45] focused on healthcare, where edge devices can be better tools for patients by directly determining their health degree. Another study [46] focused on real-time human detection and adopted a lightweight CNN deployed on the edge node and yielded satisfactory results. Some remarkable literature [43] proposed a machine learning on microcontroller unit framework (ML-MCU), which enables edge devices to train classification models with limited resources. Dedicated chips are also a viable solution [47]. It is known that most of the related works are focused on improving model recognition performance, often with model complexity increasing, but spend little attention of lightweight model and deployment on edge devices. Although some studies try to accomplish this task, some problems and limitations still exist. First, the devices used are usually the high-performance processors, which is not feasible to be widely promoted; Second, the model deployment process for edge computing needs to be improved, like the performance of embedded devices is less considered, and most of the studies only deployed the model, fail to update the model in the real-time. Third, the impact of individual differences and noise is rarely discussed. This paper proposes a low-cost universal microcontroller based on the limitations to build the real-time atrial fibrillation system. It provides a complete process for deploying an atrial fibrillation classification model on embedded devices, considering model performance and device attributes. In addition, we also introduce how the model can face the problem of individual differences. On the one hand, different thresholds can be adopted, and on the other hand, an appropriate retrain model can be adopted. Compared to the deep and huge neural network model, the model we used is relatively simple, just a three-layer forward neural network, and the related works have studied it very clearly. However, it is worth pointing out that most of them are just implemented on cloud servers, which is unsuitable for real-time, low-cost and high privacy application scenarios. Therefore, it is important to implement analysis on the edge exactly. For the insufficient resources of embedded devices, the model should be effective and lightweight, and the forward neural network is chosen for its simplicity. Taking atrial fibrillation detecting as an example in this study, we provide and present the entire process of pushing the analysis task to the edge. The main contributions of this research are as follows:

- We select suitable features to complete an AF recognition model, and the $F_1$ score is tested at approximately 90%.
- Our model is lightweight and with strong anti-noise ability, without any participation of cloud computing.
- The importance of initial models and the dataset is experimentally investigated for model retraining.

The rest of this paper is organized as follows. Section II introduces the dataset used in this paper. Section III introduces the framework of the entire system and the main function of edge devices. Section IV indicates how to design and implement this
TABLE I
DATA DESCRIPTION IN THIS STUDY

| Dataset   | Sample Frequency (Hz) | Normal | AF  | Total |
|-----------|-----------------------|--------|-----|-------|
| CinC2017  | 300Hz                 | 500    | 500 | 1000  |
| CPSC2018  | 500Hz                 | 447    | 454 | 901   |

Fig. 1. ECG recordings for atrial fibrillation and normal signals from the CinC2017 and CPSC2018 datasets.

system. Section V presents a series of progressive experimental results and discussions. Finally, Section VI briefly summarizes our work and presents some possible future directions.

II. DATA PREPARATION

The dataset used in model training is obtained from the Computing in Cardiology Challenge 2017 (CinC2017) [48]. CinC2017 dataset contains a total of 12,186 ECG recordings lasting from 9 s to 61 s. There are 8,528 recordings in the training set and 3,658 recordings in the test set. The recordings were taken by AliveCor single-channel ECG devices, which can record individuals lead I equivalent ECG and store data as 300 Hz. All recordings are classified by experts and labeled as Normal, AF, Other and Noisy. In addition, our neural network could be tested on the dataset of China Physiological Signal Challenge 2018 (CPSC2018) [49]. CPSC 2018 public training data involves 6,877 12-lead ECG recordings lasting from 6 s to 60 s. The ECG recordings were collected from 11 hospitals and sampled at 500 Hz, containing one normal type labeled as Normal and eight abnormal types, including atrial fibrillation. However, in this study, we focus on how to detect atrial fibrillation from normal ECG recordings. 500 AF-type ECG recordings and 500 normal-type ECG recordings with 30 seconds from CinC2017 are chosen, 40% for training, 40% for testing and 20% for real-time simulation. This paper selects 447 AF-type ECG recordings and 454 normal-type ECG recordings from CPSC2018 to compose another testing dataset, whose signal length is 15 seconds. The relevant information is shown in Table I.

Some representative ECG signals are presented in Fig. 1, and it is found that compared with normal signals, atrial fibrillation signals show P wave disappearance, and RR intervals in atrial fibrillation signals are more irregular than normal signals. Because ECG recordings from CinC2017 and CPSC2018 are of high quality and used features all are based on R-peak, no additional signal preprocessing is required. However, in the actual measurement process, it is recommended to use an appropriate filter to improve the signal quality, especially the impact of baseline drift and power frequency interference.

A prominent characteristic of AF is the RR interval irregularity. Because of the large amplitude and drastic changes in QRS complex slopes, AF detection based on R waves and RR intervals is appropriate. In addition, we refer to the Pan-Tompkins [50] algorithm and the Arduino heart rate analysis toolkit [51] for R wave detection. Derived the original RR interval, it is possible to calculate statistical features distinguishing AF from normal signals. Based on RRI’s irregularity theory and capabilities of embedded devices, we can calculate some heart rate variability (HRV)-based features in this paper [9], [10], [52]–[54]. The features used in this paper are defined in Table II.

In this paper, 22 common features are selected, involving time-domain analysis, frequency domain analysis and nonlinear analysis, respectively. Nevertheless, not all features can be useful, which causes feature selection to be required in machine learning.

III. APPLICATION SCENARIOS

Since the therapy development and deployment of neural networks is very mature, thus this paper does not intend to introduce it again [55]. The model parameter updating method is stochastic gradient descent with momentum (SGDM) [56]. The initial model is trained with the neural network toolbox in MATLAB, and the trained model is transplanted to edge devices. The main function is shown in Algorithm 1. The math library in the C programming language is necessary to implement it at edge devices. Fig. 2 strongly indicates that when using the unique system designed in this research, which can collect ECG signals from users, filter the raw data, recognize R-waves, extract features, run the neural network, and display their physiological.
The device can help patients know their physiological information in real-time.

The major hardware includes an Arduino microcontroller, an organic light-emitting diode (OLED), a speaker, an AD8232 integrated signal conditioning block, and a power supply. An SKX-2000sup+ ECG signal generator was used to generate ECG signals. Arduino Mega is a low-cost edge device but with extremely limited resources, where ATmega2560 MCU (8-bit, 8 KB SRAM, 4 KB EEPROM, 256 KB Flash and 16 MHz Clock) was used. This system is shown in Fig. 3. It is 101 mm long, 53 mm wide and 0.112 kg in weight. In addition, the prototype can be further simplified and more suitable for wearable usage. First, the AD8232 collects the ECG signal, and feature extraction and neural network inference are carried out inside the Arduino Mega micro-controller.

It is easy to switch to the retraining mode through the button to optimize the neural network model for a specific individual. After measuring a piece of an individualized ECG signal and extracting sufficient features for network judgment, a correct label (normal or atrial fibrillation) needs to be entered manually, which can be achieved with the help of medical practitioners. We use the ECG signal generator SKX-2000sup+ to simulate atrial fibrillation and normal ECG signals to optimize the network model. In addition to the above equipment, the rest of the hardware includes a speaker, power supply, and buttons that switch the operational modes of the system. Due to the relatively low hardware cost, this system can also serve vastly underdeveloped areas.

IV. SYSTEM DESIGN AND IMPLEMENTATION

A. Experimental Environment

The classification model runs on MATLAB and Arduino platforms. MATLAB is deployed on a personal computer equipped with an Intel (R) Core(TM) i5-9400. The edge device is an Arduino Mega microcontroller with an ATmega2560 processor, and its storage and computing capacity are inferior to those of computers. The following experiments train the initial model with MATLAB and deploy it on the edge device.

B. Classification Performance

The corresponding evaluation metrics must be built to evaluate the experimental results reasonably and objectively. Generally, for a binary classification task, there are many evaluation

```plaintext
Algorithm 1: Main Function of Edge Devices.

procedure MAIN FUNCTION (Mode)
  //Cs: Control Signal.
  //X: Patient-Specific ECG Features
  //M: Trained Model at Edge
  //Determine the current function
  if Cs==Test then
    Y ← ModelTest(M, X)
  else if Cs==Train then
    M ← ModelTrain(M, X)
  end if
end procedure

procedure MODEL TEST (M, X)
  //D: Original training Dataset.
  //Competing features normalization at first
  //Y: Predictive value
  Y ← X
  for i in M.layers do
    Y ← Y * M.weights[i]
    //Using the corresponding activation
    //Such as ReLu and tanh
    Y ← M.activation[i](Y)
  end for
  //Converts a continuous value to a category value
  //Th: Threshold, default as 0.5
  Y ← double(Y > Th)
  //Returns the current output
  return Y
end procedure

procedure MODEL TRAIN (M, X)
  //Training dataset Generate
  //Generate index randomly
  index = random(length(D))
  D(index) ← X
  //Retraining model and obtaining new parameters
  M ← update(M, D)
  Update the classification mo
  return M
end procedure
```
indexes, such as accuracy [57] and $F_1$. Accuracy can indicate how many samples are correctly distinguished, as shown in the following formula.

$$a = \frac{TP + TN}{TP + FP + TN + FN}$$

Conventionally, positive categories refer to the category with fewer samples, while negative categories refer to the category with more samples. Therefore, in this study, an ECG that indicates atrial fibrillation is regarded as a positive category, while a normal ECG is regarded as a negative category. It is difficult for a classifier always to make a correct decision, and there are four different cases in the relationship between prediction and actual value. For example, the case when the prediction and the actual value are positive is TP. The prediction precision of the positive category $p$ and recall rate $r$ are shown in the following formula.

$$p = \frac{TP}{TP + FP}$$

$$r = \frac{TP}{TP + FN}$$

Considering that both $p$ and $r$ have some defects in evaluating model, $p$ ignores the influence of FN, while $r$ ignores the influence of FP. Therefore, we use the combination of them, $F_\beta$ score, to evaluate the performance of the classifier more objectively and accurately, which is defined in the following formula. As a regulating item, $\beta$ is used to distinguish the importance of $p$ and $r$.

$$F_\beta = \frac{1 + \beta^2}{\beta^2 p + r}$$

In general, $p$ and $r$ have the same importance, that is, $\beta$ to equal one. Then, we can refer to $F_\beta$ as the $F_1$, which is more suitable for evaluating model performance with extremely unbalanced data.

### C. System Implementation

In this section, the experimental setup is introduced. First, we conduct a preliminary experiment to select features for AF detection. The following experiments include initial model training, real-time edge inference simulations, and model retraining simulations. When we train the initial model, it is important that make sure the number of normal ECG signals should equal that of AF signals, which can effectively rule out the effect of data imbalance. Repetitive experiments have been completed to obtain general results. The experiment on real-time edge inference verifies the model performance in real scenarios, laying a foundation for further research on model updating. Inference on edge devices is a necessary condition for real-time retraining. Therefore, it is necessary to retrain the model under a realistic simulation of our system.

1) **Feature Selection**: It is difficult to guarantee that the features extracted in the previous section are beneficial for a specific task [2]. Therefore, the main purpose of this experiment is to get the optimal subset from some general features. The process of feature selection is shown in Algorithm 2.

#### Algorithm 2: Feature Selection Process

// $FS$: The set of features.
// $Num$: The number of features.

```
for i in $Num$
do
    // Estimating its computational complexity.
    if the computational complexity is high then
        $FS(i)$ ← Null
    end if
    if $FS(i)$ with weak label correlation then
        if $FS(i)$ with small variance then
            $FS(i)$ ← Null
        end if
    end if
end for
```

// Calculating Pearson correlation in $FS$
// Choosing one feature in co-linear features
// Obtaining the current features subset $FS$

Relevant research may mainly focus on improving model performance [58]. However, this paper considers the factor and impact introduced by embedded devices. In feature selection, three factors are selected: characterization ability (informative enough), strong correlation with labels (and weak correlation within features) and low computational complexity. The first two factors are easily found in relevant research, while low computational complexity is rarely mentioned due to abundant computing resources. However, computational complexity should be seriously considered in the case of edge devices with limited computing resources. There are three main processes in calculating correlation features: multiplication (or division), addition (or subtraction) and value judgement. Next, we calculate the variance and label correlation of each feature. The variance is calculated as follows, where $t_{ij}$ represents the value of the $j$th feature in the $i$th sample, assuming there are M samples.

$$t_{ij} = \frac{1}{M} \sum_{i=1}^{M} t_{ij}$$

$$v_j = \frac{1}{M} \sum_{i=1}^{M} (t_{ij} - \bar{t}_j)^2$$

The label correlation is the information entropy gain obtained by finding the most appropriate value for distinguishing AF from normal signals in each feature, and this idea is from the decision tree [59]. First, we define information entropy $E$ in the following equation. Then we can find an optimal segmentation point for each feature to increase the purity. By calculating the initial information entropy $E_0$ and information entropy in the $j$th feature $E_j$, we can obtain label correlation of all features, as shown in the following formula.

$$E = -(p_a \log(p_a) + p_{af} \log(p_{af}))$$

$$g_j = E_0 - E_j$$
used to measure the importance of the \(j\)th feature. The higher the value, the more suitable the feature is for classification.

2) Initial Model Training: Previously, we selected the required features through some criteria, and the features will be used in model training to verify the feasibility. Here, the training process of the initial model was mainly completed on MATLAB, and the termination condition was that the epoch reaches 5000, which uses a 5-3-1 three-layer neural network model structure. Before this experiment, the pre-experiment was conducted to verify network structures by changing the values of hidden layer neurons \(numh\). The learning rate was set to 0.2, the momentum value was 0.9, and the momentum combined with the gradient descent method was used to train the model. The model is tested for performance on two text sets. The post-processing threshold is set to 0.5. Based on this experiment, the model structure is feasible, and the feature selection is correct.

3) Real-Time Inference Simulation: In addition to the model’s performance, we are also concerned about the deployment of the model and real-time application. However, due to the lack of enough clinical ECG signals, we construct a real-time simulation dataset. In this section, we use this dataset for real-time inference simulation, and in the next experiment, we also use this dataset for model retraining. The system we designed is shown in Fig. 3. An ECG is collected through three electrodes, and the analysis process is fully completed by this system, leading the user to obtain results in real-time. If abnormal signals are detected, the system can save these signals for further judgment with the assistance of doctors. Moreover, the system can be further simplified to increase its portability. This section will discuss the influence of the post-processing threshold on real-time classification results. Setting the threshold can improve the model’s performance and adaptability. Due to the influence of the other physiological signals and environment, noise’s role in ECG signals cannot be ignored. Here, we will use the concept of signal-to-noise ratio (SNR), which is defined as the ratio of the power of the signal \(P_s\) to the power of the noise \(P_n\), expressed in decibels.

\[ SNR = 10 \log \frac{P_s}{P_n} \] (9)

Lastly, response time is an extremely important factor, and the device should be able to respond to the user in a much shorter time, and the signal length is a fixed time. This section compares and analyzes different data processing modes from four aspects: data source, data processing location, whether it is wearable and response time.

4) Model Retraining: If the device fails to judge the target data from a specific user, which results in some serious consequences, it is of great importance to retrain and update the initial model for users. The model performs poorly on some patterns that it has not observed before. Previous experimental results show a pattern: samples that fail to be recognized are highly overlapped in parallel experiments. The training dataset mentioned earlier is initial, for it can be updated in real-time. This research uses a combination of old and new data rather than just new data because we want the model to cover more patterns. To study the real-time training model of embedded devices, we specifically design the following experiment. The experiment requires that the new input data replace a sample in the training dataset, which is used as the category replacing (CR) method. The \(F_1\) score is chosen as the evaluation metric in this experiment.

The numbers of normal and AF signals in both datasets are equal. Three groups are built based on group N (original model, which can be a baseline.), involving group A (without the initial dataset and trained model), group B (without the initial dataset but with the trained model), and group C (with the initial dataset and trained model).

V. RESULTS AND DISCUSSIONS

A. Feature Selecting

Since a single multiplication (division) takes more time than addition (subtraction), the number of multiplications (division) in the feature calculation is considered, and the space required for each feature is also taken into account. Our estimation results on the computational complexity of related features are shown in Table IV.

The computing complexity of frequency-domain features is much higher than that of time-domain features and several nonlinear features. In addition, frequency-domain features require more storage memory due to the interpolation of RR intervals. Therefore, frequency-domain features are not recommended. Variance and label correlation can be obtained for each feature. The normalized values are shown in Fig. 4.
The variance and label correlation of features; the graph on the left represents the variance of the features of the training dataset, while the graph on the right shows the correlation between features and labels.

Under the constraint of computational complexity, the features we need are rich information and high label correlation. The first indicator only considers features’ attributes, which is like unsupervised learning. While the second indicator is supervised learning. In addition, the strongly correlated features are discarded. Linear correlation analysis for all features is shown in Fig. 5.

Firstly, we pre-defined the size of the feature subset for model input as 5. The frequency-domain features are excluded for the high computing complexity according to Table IV. Based on Fig. 4, NN50, PNN50, NN20 and PNN20 rank top, but these four features are highly correlated. Moreover, NN20 and NN50 are influenced by the signal length. Therefore, PNN50 was selected as one of the features. For ranking fifth in label correlation, CovRR was also selected. In terms of high variance, SDSD and MRR were selected. SKEW was chosen for its weak correlation with other features. The collinearity of selected features is relatively weak, and the maximum is 0.6878. The following experiment is conducted with these features.

By the knowledge brought by features, it avoids the need for enormous data volume and large models, making it more suitable for deployment on edge devices. However, something should be noticed that only features based on heart rate variability are chosen and analyzed, and other features may be helpful in classification. Analyzing more features is helpful to find the most suitable feature subset for a special task. This paper mainly shows the idea of feature selection. It is true that manual features limit the model’s performance to some extent, but the feature-based models can perform well with fewer parameters.

B. Initial Model Training

The test results are shown in Fig. 6. CinC2017 belongs to Test1, while CPSC2018 belongs to Test2, as most models have approximately 90% accuracy. The error bars indicate that the training process has good repeatability and further prove feature engineering feasibility. It costs about 3 seconds to train the initial model on MATLAB. In addition, just one model is needed in the actual deployment. The most prominent model is selected based on the $F_1$ above Test1, and these confusion matrixes are shown in Fig. 7.

As seen from the confusion matrix, the model we built can obtain 93% accuracy in Test1 and 94.5% accuracy in Test2, presenting outstanding classification ability. In addition, we can also notice that the selection of features is of great importance, and the relevant results are shown in Table V. Based on the $F_1$-score, it is known that the model trained with the selected features performs better than the model trained with all features in the test dataset, especially CPSC2018.
Moreover, both the input and the model are more lightweight to be implemented on embedded platforms. The model with all features structure has 73 parameters, while the model with selected features has 22 parameters. However, as mentioned above, the interpolation process needs to be carried out in advance to calculate the frequency domain features, leading to the input size will be much larger than that of the model with selected features. The structure considered in this paper is a multi-layer perceptron, and the 5-3-1 structure was adopted through structure search. It requires a total of 18 multiplications and 18 additions. The low computational complexity makes running the model at the edge device possible. However, other machine learning models are not studied, which may have better model performance.

C. Real-Time Inference Simulation

As shown in Fig. 3, the system was built, and then we presented the actual working process of the system, and a signal from the ECG simulator was recorded, as shown in Fig. 8.

The ECG simulator generates four ECG recordings involving two normal signals and two atrial fibrillation signals, all of which can be correctly classified on the embedded device. In addition, a normal person with no history of cardiovascular-related was tested using the system that is able to judge these recordings as normal. However, the amount of ECG signals in the real-time scenario is limited, especially abnormal recordings. In the next experiment, we will use the real-time simulation dataset. Firstly, the impact of the threshold should be discussed. It is known that a continuous value can be calculated by this model, but the actual output is 0/1, which represents normal or atrial fibrillation. Generally, researchers will carry out the threshold processing, and the default threshold we used in this paper is 0.5. However, in this real-time simulation, we want to adjust the threshold value according to the needs of users. For example, for some potential patients, we can lower the threshold appropriately. The effects of different thresholds under the simulation data set are shown in Fig. 9.

According to the above experimental results, with a small threshold, there is fewer false-negative at this time, resulting in the model performing perfectly in Recall. As the threshold increase, Recall goes down while Precision goes up. Therefore, it is not recommended to use the two indicators alone in this paper. The model performance can be evaluated by $\text{F}_{1}$ (the special form is $F_1$). For example, in the above experimental results, $F_1$ can reach 0.9045 when the threshold value is 0.7. Using this threshold, we change the SNR value, and the corresponding model performance is shown in Fig. 10.

The experimental results show that our method has a certain anti-noise ability. This phenomenon is because R-peaks are relatively not easily disturbed by noise, leading to the realization of high accuracy.
that atrial fibrillation identification based on R-peaks is suitable for deployment in practical application scenarios. Next, this section will take a glance at the meanings and advantages of edge computing, such as low response time and low power consumption. For example, response time includes transmission time and computing time. Based on the same transmission method, the closer the data source is, the shorter the transmission time. The computing time mainly depends on the computing capacity of the device. To illustrate the advantages of edge computing, we designed the following experiment. In this experiment, the control group is the edge devices that collect ECG signals and send the original signal to a computer for processing through a serial port (or other communication approaches). The experiment is conducted with 150 samples from the simulation dataset, and the response time that does not include the time of signal collecting and feature extracting is computed. The experimental results are shown in Table VI.

In Mode 1, data are stored on the server, so data transmission is not required. However, it is difficult to implement this in wearables, because it requires a direct connection to the server. Mode 2 requires the embedded devices to collect data and then send it to the server for processing, so its response time is the sum of the calculation time and transmission time. The first two modes require the server’s participation and high network bandwidth dependency. Mode 3 is proposed for these limitations and problems. By comparing these data processing modes, some conclusions are obtained: firstly, the server computing speed is much faster than that of edge devices, and the server is capable of parallel computation. If the transmission process is considered, the direct processing on the edge devices can obtain lower latency and lower power consumption. More importantly, the risk of privacy leakage can be effectively eliminated. The results of Table VI indicate the feasibility and some advantages of edge inference, which is the basis for the further design of individualized lightweight neural networks for the edge device. However, the data used in this experiment is not real-time data. Therefore, we should try to obtain data from clinical sources in the future. In addition, it should be pointed out that edge computing is a supplement to cloud computing, rather than a substitute. However, in this paper, we mainly focus on and discuss edge computing, and do not discuss edge-cloud collaboration, which also needs to be supplemented.

D. Model Retraining

In order to explore how the initial training set and the pre-training model affect the performance of the new model, we set up the following three groups of experiments. The relevant experimental results are shown in Table VII, which describes a simple case. Group N trains the initial model with 100 recordings, and groups A, B, and C train the model with new 20 recordings in a real-time simulation dataset. Therefore, there is not much difference between groups.

It is known that models based on trained models generally improve training speed, which has been studied in the field of transfer learning [60]. We also need to study the impact of the total number and proportion of new data on the model performance. Considering clinical practice scenarios, ECG from a user is not necessarily homogeneous, and the number of new data should not have a limitation. To control variables and contrast with Table VII, normal and AF signals are evenly distributed when the total number changes. In the case of varying proportions, we show the experimental results of 20 samples. Considering the imbalanced data, we select the $F_1$ score to present the results.

The new dataset is a part of the real-time simulation dataset. Group N is used for comparison, and the pre-trained model can play a very important role. The experimental results are even more pronounced when the sample size is 10. The performance of group B and group C can surpass group A in the same iterations due to the pre-trained model.

At this point, the ratio of atrial fibrillation to normal signal in the new dataset is 1/9, which is the greatest observed difference between signals. Although group B contains the original model, it still performed poorly, proving the significance of keeping the original dataset for model retraining, even with its space consumption in embedded devices.

VI. SUMMARY

This paper has designed an edge AF detection system based on the embedded platform that mainly uses feature engineering to collect ECG signals in real-time and detect atrial fibrillation. The model used for atrial fibrillation is lightweight level, with
22 parameters and the F1-score is about 90%. It only takes 18 multiplications and 18 additions. In addition, based on the comparison of data processing modes, it is known that directly analyzing data at the edge can reduce the response time. The edge computing framework in this paper has many advantages, and the entire process can turn to other tasks. However, this study lacks clinical data verification, which needs the participation of hospitals and patients. Cooperating with hospitals, realizing multi-classification, and managing edge-cloud will be further researched in the future. These devices can be implemented in areas with limited healthcare resources, such as rural areas. These patients can then be sent to the hospital for timely diagnosis and immediate treatment.

REFERENCES

[1] L. Zhu et al., “Atrial fibrillation detection and atrial fibrillation burden estimation via wearables,” IEEE J. Biomed. Health Inform., vol. 26, no. 5, pp. 2063–2074, 2022, doi: 10.1109/JBHI.2021.3131984.

[2] A. B. We et al., “Accurate detection of atrial fibrillation from 12-lead ECG using deep neural network,” Comput. Biol. Med., vol. 116, 2020, Art. no. 10378.

[3] L. Friberg, M. Rosenqvist, A. Lindgren, A. Ternt, and K. Asplund, “High prevalence of atrial fibrillation among patients with ischemic stroke,” Eur. J. Neurol., vol. 22, no. 9, 2014, Art. no. 645.e5.

[4] S. Nattel, “New ideas about atrial fibrillation 50 years on,” Nature, vol. 415, no. 6868, pp. 219–226, 2002.

[5] C. Yao et al., “Classification of short single lead electrocardiograms (ECGs) for atrial fibrillation detection using piecewise linear spline and Xgboost,” Physiol. Meas., vol. 39, no. 10, 2018, Art. no. 104006.

[6] X. Du, L. Guo, S. Xia, J. Du, and C. Ma, “Atrial fibrillation prevalence, awareness and management in a nationwide survey of adults in China,” Heart, vol. 107, no. 7, pp. 535–541, 2021.

[7] F. Witassek, A. Springer, L. Adam, S. Aeschbacher, and M. Schwengerlnks, “Health-related quality of life in patients with atrial fibrillation: The role of symptoms, comorbidities, and the type of atrial fibrillation,” PLoS One, vol. 14, no. 12, 2019, Art. no. e0226730.

[8] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun, “Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review,” Comput. Biol. Med., vol. 122, no. 5, 2020, Art. no. 103801.

[9] H. Yuki et al., “Computer-aided diagnosis of atrial fibrillation based on ECG signals: A review,” Int. Sci., vol. 467, pp. 99–114, 2018.

[10] A. Riswan, A. Zoha, I. B. Mabrouk, H. M. Sabour, and Q. H. Abbasi, “A review on the state of the art in atrial fibrillation detection enabled by machine learning,” IEEE Trans. Biomed. Eng., vol. 14, pp. 219–239, 2021.

[11] H. Holst, M. Ohlsson, C. Peterson, and L. Edenbrandt, “A confident decision support system for interpreting electrocardiograms,” Clin. Physiol. Physiol., vol. 19, no. 5, pp. 410–418, 2010.

[12] M. J. M. Carlos et al., “An open IoT-based deep learning framework for online medical image recognition,” IEEE J. Sel. Areas Commun., vol. 39, no. 2, pp. 541–548, Feb. 2021.

[13] S. Kiranyaz, T. Ince, and M. Gabbouj, “Real-time patient-specific ECG classification by 1-D convolutional neural networks,” IEEE Trans. Biomed. Eng., vol. 63, no. 3, pp. 664–675, Mar. 2016.

[14] Z. Xiong, M. K. Stiles, and J. Zhao, “Robust ECG signal classification from a single lead,” IEEE J. Biomed. Health Inform., vol. 25, no. 7, pp. 2758–2767, Jul. 2021.

[15] M. Naz, J. H. Shah, M. A. Khan, and M. Sharif, “Mudassar Raza, and Robertas Damasevicius,” from ECG signals to images: A transformation based approach for deep learning,” PeerJ Comput. Sci., vol. 7, 2021, Art. no. e3586.

[16] B. Fang et al., “Dual-channel neural network for atrial fibrillation detection from a single lead ECG wave,” IEEE J. Biomed. Health Informat., to be published, doi: 10.1109/JBHI.2021.320890.

[17] M. Limam and F. Precioso, “AF detection and ECG classification based on convolutional recurrent neural networks,” in Proc. Comput. Cardiol., 2017, pp. 1–4.

[18] 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,” Inf. Fusion, vol. 53, pp. 194–1–200, 2020.

[19] J. Cai, W. Sun, J. Guan, and I. You, “Multi-ECGNet for ECG arrhythmia multi-label classification,” IEEE Access, vol. 8, pp. 110848–110858, 2020.

[20] T. Radhakrishnan et al., “AFCNNNet: Automated detection of AF using chirplet transform and deep convolutional bidirectional long short term memory network with ECG signals,” Comput. Biol. Med., vol. 137, 2021, Art. no. 104783.

[21] C. A. Gomez-Garcia, M. A. Askar-Rodriguez, and J. Velasco-Medina, “Platform for healthcare promotion and cardiovascular disease prevention,” IEEE J. Biomed. Health Informat., vol. 25, no. 7, pp. 2758–2767, Jul. 2021.

[22] A. G. Marcus et al., “Online heart monitoring systems on the internet of health things environments: A survey, a reference model and an outlook,” Inf. Fusion, vol. 53, pp. 222–239, 2020.

[23] O. Lahdenoja et al., “Atrial fibrillation detection via accelerometer and gyroscope of a smartphone,” IEEE J. Biomed. Health Informat., vol. 22, no. 1, pp. 108–118, Jan. 2018.

[24] Y. Liu, B. Fang, Y. Zhao, and J. Chen, “Ensemble learning for atrial fibrillation screening from a single lead ECG wave of wearable devices,” in Proc. IEEE 3rd Int. Conf. Front. Technol. Inf. Comput., 2021, pp. 590–594.

[25] Y.-J. Lin, C.-W. Chuang, C.-Y. Yen, S.-H. Huang, J.-Y. Chen, and S.-Y. Lee, “Live demonstration: An aiot wearable ECG patch with decision tree for arrhythmia analysis,” in Proc. IEEE Biomed. Circuits Syst. Conf., 2019, pp. 1–4.

[26] H. D. M. Ribeiro et al., “ECG-based real-time arrhythmia monitoring using quantized deep neural networks: A feasibility study,” Comput. Biol. Med., vol. 143, 2022, Art. no. 105249.

[27] L. Xiaolin, C. Rajesh, B. P. Cardiff, and D. John, “Multistage pruning of CNN based ECG classifiers for edge devices,” in Proc. 43rd Ann. Int. Conf. IEEE Eng. Med. Biol. Soc., 2021, pp. 1965–1968.

[28] A. Uki et al., “Resource constrained CVD classification using single lead ECG on wearable and implantable devices,” in Proc. 43rd Ann. Int. Conf. IEEE Eng. Med. Biol. Soc., 2021, pp. 886–889.

[29] D. Liang, T. Wong, Y. Li, D. John, W. K. Ho, and C. H. Heng, “Resource and energy efficient implementation of ECG classifier using binary CNN for edge AI devices,” in Proc. IEEE Int. Symp. Circuits Syst., 2021, pp. 1–5.

[30] Z. Wang et al., “A 148-nm reconfigurable event-driven intelligent wake-up system for AoIoT nodes using an asynchronous pulse-based feature extractor and a convolutional neural network,” IEEE J. Solid-State Circuits, vol. 56, no. 11, pp. 3274–3288, Nov. 2021.

[31] B. Sadharsan, J. G. Breslin, and M. L. Ali, “Edge2Train: A framework to train machine learning models (SVMS) on resource-constrained IoT edge devices,” in Proc. 10th Int. Conf. Internet Things, 2020, pp. 1–8.

[32] A. John, C. Rajesh Panicker, B. Cardiff, Y. Lian, and D. John, “Binary classifiers for data integrity detection in wearable IoT edge devices,” IEEE Open J. Circuits Syst., vol. 1, pp. 88–99, 2020.

[33] M. Caprolu, R. D. Pietro, F. Lombardi, S. Raponi, and M. Z. Delai, “A novel AIoT platform for atrial fibrillation recognition and detection,” in Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst., 2018, pp. 852–859.
[41] M. Alrowaily and L. Zhuo, “Secure edge computing in IoT systems: Review and case studies,” in Proc. IEEE/ACM Symp. Edge Comput., 2018, pp. 440–444.
[42] J. Xiao, Q. Liu, H. Yang, J. Liu, and J. Zhou, “ULECGNet: An ultra-lightweight end-to-end ECG classification neural network,” IEEE J. Biomed. Health Inform., vol. 26, no. 1, pp. 206–217, Jan. 2022.
[43] B. Sudharsan, J. G. Breslin, and M. I. Ali, “ML-MCU: A framework to train ML classifiers on MCU-based IoT edge devices,” IEEE Internet Things J., to be published, doi: 10.1109/JIOT.2021.3098166.
[44] F. Alongi, N. Ghielmetti, D. Pau, F. Terraneo, and W. Fornaciari, “Tiny neural networks for environmental predictions: An integrated approach with Miosix,” in Proc. IEEE Int. Conf. Smart Comput., 2020, pp. 350–355.
[45] T. Reddy, G. C. Sirisha, and A. M. Reddy, “Smart healthcare analysis and therapy for voice disorder using cloud and edge computing,” in Proc. 4th Int. Conf. Appl. Theor. Comput. Commun. Technol., 2018, pp. 103–106.
[46] S. Y. Nikouei, C. Yu, S. Song, R. Xu, and T. R. Faughnan, “Real-time human detection as an edge service enabled by a lightweight CNN,” in Proc. IEEE Int. Conf. Edge Comput., 2018, pp. 125–129.
[47] W.-C. Fang, K.-Y. Wang, N. Fahier, Y.-L. Ho, and Y.-D. Huang, “Development and validation of an EEG-based real-time emotion recognition system using edge AI computing platform with convolutional neural network system-on-chip design,” IEEE Trans. Emerg. Sel. Topics Circuits Syst., vol. 9, no. 4, pp. 645–657, Dec. 2019.
[48] D. Gari et al., “AF classification from a short single lead ECG recording: The phisioNet/Computing in cardiology challenge,” in Proc. Comput. Cardiol., 2017, pp. 1–4.
[49] F. Liu, C. Liu, L. Zhao, X. Zhang, and E. N. Y. Kwee, “An open access database for evaluating the algorithms of electrocardiogram rhythm and morphology abnormality detection,” J. Med. Imag. Health Inform., vol. 8, no. 7, pp. 1368–1373, 2018.
[50] J. Pan and W. J. Tompkins, “A real-time QRS detection algorithm,” IEEE Trans. Biomed. Eng., vol. BME-32, no. 3, pp. 230–236, Mar. 1985.
[51] P. V. Gent, H. Farah, N. V. Nes, and B. V. Are, “Heart rate analysis for human factors: Development and validation of an open source toolkit for noisy naturalistic heart rate data,” in Proc. HUMANIST Conf., 2018, pp. 173–178.
[52] A. J. Camm et al., “Heart rate variability standards of measurement, physiological interpretation, and clinical use task force of the European society of cardiology and the North American society of pacing and electrophysiology,” Circulation, vol. 93, no. 5, pp. 1043–1065, 1996.
[53] I. Cygankiewicz and W. Zareba, “Heart rate variability,” Handbook Clin. Neurol., vol. 117, no. 4, pp. 379–393, 2013.
[54] F. Shaffer and J. P. Ginsberg, “An overview of heart rate variability metrics and norms,” Front. Public Health, vol. 5, 2017, Art. no. 258.
[55] A. Blum, Neural Networks in C++ an Object-Oriented Framework for Building Connectionist Systems. Hoboken, NJ, USA: Wiley, 1992.
[56] A. Cutkosky and H. Mehta, “Momentum improves normalized SGD,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 2260–2268.
[57] C. Ma, S. Wei, T. Chen, J. Zhong, and C. Liu, “Integration of results from convolutional neural network in a support vector machine for the detection of atrial fibrillation,” IEEE Trans. Instrum. Meas., vol. 70, pp. 1–10, 2021.
[58] R. Mahajan, R. Kamaleswaran, and O. Akbilgic, “A hybrid feature extraction method to detect atrial fibrillation from single lead ECG recording,” in Proc. IEEE EMBS Int. Conf. Biomed. HealthInformat., 2018, pp. 116–119.
[59] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” IEEE Trans. Syst., Man, Cybern., vol. 21, no. 3, pp. 660–674, May/June 1991.
[60] S. J. Pan and Y. Qiang, “A survey on transfer learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010.