Detection of Atrial Fibrillation Using 1-D Convolutional Neural Network

Chaur-Heh Hsieh[1], Yan-Shuo Li[2], Bor-Jiunn Hwang[2], and Ching-Hua Hsiao[2]

[1]College of Artificial Intelligence, Yango University, Fuzhou, China;
[2]Department of Computer and Communication Engineering, Ming Chuan University, Taoyuan, Taiwan

***Abstract*—The automatic detection of AF is crucial for its association with the risk of embolic stroke. Most of existing AF detection methods usually convert 1-D time-series ECG signal into 2-D spectrogram to train a complex AF detection system, which result in heavy training computation and high implementation cost. This paper proposes an AF detection method based on end-to-end 1-D CNN architecture to raise the detection accuracy and reduce network complexity. By investigating the impact of major components of a convolutional block on detection accuracy and using grid search to obtain optimal hyperparameters of CNN, we develop a simple, yet, effective 1-D CNN. Since the dataset provided by Physionet challenge 2017 contains ECG recordings with different lengths, we also propose a length normalization algorithm to generate equal-length records to meet the requirement of CNN. Experimental results and analysis indicate that our method of 1-D CNN achieves better detection accuracy with lower network complexity, as compared with the existing deep-learning-based methods.**

***Index Terms*— Electrocardiogram (ECG), Atrial Fibrillation (AF), Convolutional Neural Network (CNN), Deep learning**

I. Introduction

An arrhythmia is an irregular heart beating problem. A common type of arrhythmia is atrial fibrillation (AF) caused by the abnormal of sinus rhythm and irregular heartbeat which could lead to complications such as heart attack [1]. According to a report from the National Institute of Health, there are 2.2 million Americans suffering from AF and the possibility of having a stroke increases as aging. A recent investigation [2] showed that people in the UK were diagnosed with heart failure on an increase of 12%. These increasing heart diseases problems lead to heavy financial burden among countries. A study [3] showed that the average economic cost of heart failure was estimated about USD 108 billion in 197 countries. Due to the increase of atrial fibrillation, development of an effective automatic detection method for AF from electrocardiogram (ECG) signals is valuable for healthcare. However, so far it is still a challenging task for computers due to its episodic nature [4],[5].

ECG is a medical signal which tests the abnormality of heart by measuring the electrical activity of the heart. A beat of ECG signals can be observed by five characteristic waves－P, Q, R, S, and T, as shown in Figure 1 [6]. P wave stands for depolarization; QRS complex corresponds to the depolarization of the right and left ventricles, and contraction of the ventricular muscles; T wave represents repolarization of the ventricles.

The characteristics of AF can be diagnosed by ECG, including no clear P-wave, abnormal pattern of R-R interval, irregularity of heart beating, and small oscillation (frequency 4 -8Hz) of ECG baseline.

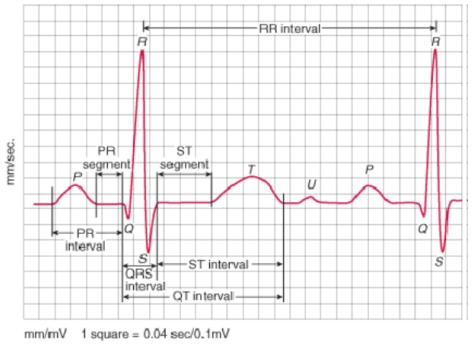

Fig. 1. Electrocardiogram

The abnormality of the ECG signal is generally identified by using classification techniques [7], which are achieved by extracting the features of the ECG that can discriminate the defined categories. Conventionally, the features are derived from the magnitude, duration and area of the QRS, T and possibly P waves [8].

In order to improve the performance of detection and classification of ECG signals, several analysis methods based on machine learning classifiers have been developed such as particle swarm optimization (PSO), support vector machines (SVM), self-organizing map (SOM), fuzzy c-means (FCM) clustering, rough sets, k-nearest neighbor (KNN), artificial neural network (ANN) classifier, and quantum neural network (QNN) based classifier [8]-[18]. For examples, in [8], rough sets (RS) and QNN are employed to identify ECG signals. In [10], the QRS complex is described by Hermite polynomials, and the coefficients of the polynomial are fed into the neuron-fuzzy classifier. In [13], a method using feature selection driven by database generalization criteria is proposed for heartbeat classification. In [14], the authors propose an arrhythmia identification method based on NSVM–KNN hybrid classifier. Using multi-resolution wavelet transform (WT) and ANN classifier for abnormality detection is presented in [15].

In [19], a two-layer cascaded binary classifier is proposed to make four-label classification. The ECG recording is classified into one of two classes at the first layer and classified separately

at the second layer. Pan-Tompkins algorithm is utilized to extract features of a recording. Adaboost, an ensemble learning approach, is presented to reduce the misclassification rate.

In [20], the authors make use of QRS complex features to train a classifier based on long short-term memory (LSTM). A modification of the Pan-Tompkins R peak detection algorithm is developed to detect R peak for further feature extraction. A total of 23 features, which include R-R interval, R amplitude difference, heart rate turbulence offset, etc., are extracted from signals. A sequence of features divided into chunks of 46 time steps are fed into 3-layer LSTM network. The advantage of the method is that it uses deep architecture with less training parameters. However, the accuracy of this method is driven by R peak detection since the derived features would be inaccurate if the R peak detection is inaccurate. The method in [21] is also based on features derived from R-R intervals. These features are processed with a quadratic neural network classifier. Another R peak-based scheme using shallow convolutional neural network (CNN) structure is presented in [22]. Similar to the previous work, the accuracy of these methods is also limited by R peak detection.

The methods stated above basically belong to conventional machine learning (ML). That is, they first obtain hand-crafted features through conventional feature extraction and selection algorithms, and then train a classification model using these features. The quality of extracted features significantly affects performance of the detection/classification. However, the hand-crafted features are not generally robust with respect to many variations, such as scaling, noise, displacement, etc. Recently, deep learning (DL) has achieved great success in various problems, especially in computer vision and speech recognition. Several researchers have applied DL frameworks such as stacked auto-encoder, CNNs and LSTMs for the analysis of ECG signals [23]-[27].

Luo, et al. [23] presents a patient-specific DNN heartbeat classifier. A stacked denoising auto-encoder (SDA) is pretrained by a time-frequency spectrogram obtained with unlabeled modified frequency slice wavelet transform. Then, a deep neural network model is initialized by the parameters of the trained SDA and then updated with fine-tuning.

In [24], the authors apply the approach in [25] which is constructed with 34-layer 1D ResNet for the classification of Physionet challenge dataset. They also slightly modify the ResNet by reducing number of filters per convolutional layer and using different input segment lengths. For simplicity we referred the two methods as ResNet-1 and ResNet-2, respectively.

Warrick et al. [26] propose a CL3 network for ECG classification that consists of two learning stages: representation learning and sequence learning. The former employs 1-D CNN to extract local and discriminative features of input signals; the latter is implemented with three stacked LSTM layers to learn long term features.

In [27], the authors propose a method which applies the spectrogram of ECG signal to train on a convolutional recurrent neural network (CRNN). This method firstly transforms raw ECG data into 2-D spectrogram using Fast Fourier Transform. The spectrogram is fed into a stack of 2-D convolutional blocks and then a 3-layer bidirectional LSTM network is followed for feature aggregation. However, the network complexity is rather high with total number of training parameters of more than 10 million.

In summary, the existing ML-based methods utilize expert knowledge to extract features, which is time-consuming and error-prone. The DL-based methods can be categorized into 1-D and 2-D schemes. In 1-D scheme, the time-series 1-D data is directly fed into neural network. On the contrary, for 2-D scheme, the time-series signal should be converted into 2-D form, i.e. spectrogram, before inputting the subsequent neural network. Compared to 1-D schemes, the 2-D schemes improve the classification accuracy around 3 % [24],[26],[27] but at the cost including (a) overhead of conversion, and (b) increasing the network complexity.

In order to raise the detection accuracy and reduce network complexity, this paper proposes an AF detection method based on an end-to-end 1-D CNN architecture. By investigating the impact of major components of a convolutional block on detection accuracy and using grid search to obtain optimal hyperparameters of CNN, we develop a simple, yet effective 1-D CNN. The 1-D time-series raw data is fed into the network without any conversion. In addition, this AF detection is data driven, i.e., the features of ECG signal and classification model are learnt from raw data directly. Since the dataset provided by Physionet challenge 2017 contains ECG recordings with different lengths, we also propose a length normalization algorithm to generate equal-length records to meet the requirement of CNN.

Conventionally, prediction accuracy (detection or classification accuracy) is often used as a metric for the evaluation of DNN-based classifier. However, the complexity of DNN is also important since it will affect the computational load and implementation cost of a system. Thus, in this paper we use prediction accuracy and network complexity, presented in our previous work [28], as the metrics for the evaluation and comparison of the deep neural networks. The major contributions of this work are summarized as:

(a) We propose a simple, yet, effective AF detection method based on end-to-end 1-D CNN architecture to classify time-series 1-D ECG signal directly without converting it to 2-D data. Through exhaustive evaluation, we prove our method achieves better detection accuracy than the existing DL-based methods. In addition, the proposed method reduces network complexity significantly, as compared with the second-ranked method CRNN.
(b) We study the effect of the batch normalization and pooling methods on detection accuracy, and then design the best network by combing the grid search method.
(c) We present a length normalization algorithm to solve variable length of ECG recordings.

The remainder of this paper is organized as follows. Section II first gives an overview of the proposed method, describes the data pre-processing algorithm, and then the design of 1-D CNN deep neural networks. The numerical analysis and performance comparison are given in Section III. Finally, the conclusion is drawn in Section IV.

II. Proposed AF Detection

A. System Overview

The proposed AF detection system shown in Figure 2 includes raw data pre-processing, offline training and online prediction. The length of ECG recordings of the dataset we used is variable, which is not suitable for the deep neural network. Therefore, we develop a pre-processing algorithm to make each recording fixed in length. The dataset contains four classes of ECG signals: AF, Normal, Noisy, and Other. The detection of AF can be easily formulated as a classification problem. Thus, we design a 1-D CNN for the classification of the ECG signals. The classifier design consists of training and inference. In training phase, the 1-D CNN predicts one of the 4 classes for training data. The loss (error) between predicted output and ground truth output is calculated and then back-propagates to each layer of the network so as to update the parameters of the network iteratively. An optimal network model is obtained when the iteration process converges. In inference phase, the test data is applied to the 1-D CNN with the optimal model, and the predicted result is generated.

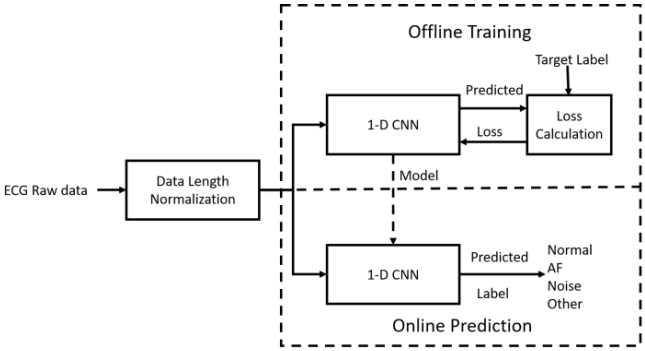

Fig. 2. Flowchart of the proposed AF detection method

B. Data Length Normalization

The dataset we used is provided by Physionet challenge 2017, containing 8,528 single lead variable-length ECG recordings lasting from 9 seconds to 61 seconds. The recordings are sampled at 300 Hz and has been bandpass filtered by the AliveCor KardiaMobile device. The dataset is composed of four classes, including AF (Atrial Fibrillation), Normal (Normal sinus rhythm), Noisy (too noisy to be recognized), and Other (Other rhythm), which was annotated by experienced experts.

Due to the difficulty of putting variable-length data into deep neural network, length normalization which makes each recording into a segment with fixed length should be performed first. The length normalization problem is rather common across many domains. In [24],[29] the authors manually preset a required length (called length threshold here), and then cut the original data into segments with the required length. The method is rather simple, but its major problem is that the setting of the length threshold is done manually, which does not consider the characteristic of the dataset. Setting length threshold to any value in the range of the recording lengths (9 seconds to 61 seconds) will generate bias. The problem is how to minimize the bias by generating an appropriate length threshold in an automatic mechanism.

In this work, we develop a histogram-based length normalization algorithm, which automatically determine the length threshold value using the histogram distribution of the length of the recordings. The histogram of the lengths of 8,528 recordings is shown in Figure 3. It indicates the length of the majority of data fall in the range around 30 seconds. Therefore, in our length normalization algorithm, the length threshold is set to 30 seconds, which corresponds to 9,000 samples since sampling rate is 300 samples per second. The algorithm will make the least bias in statistical sense since it considers the most frequent data length as a threshold.

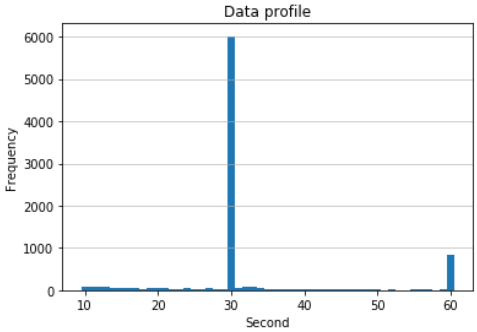

Fig. 3. Data length histogram distribution

The normalization algorithm not only solves the variable-length data problem but generates more data to train deep neural network. The pseudocode of the algorithm is shown in Algorithm 1. If the recording contains just 9,000 samples (30 seconds), this recording will be used directly without any modification. If the recording contains more than 9,000 samples, multiple segments will be generated with 50% overlap between adjacent segments. Each segment from the same recording has 9,000 samples and will be assigned the same label. For instance, an AF recording containing 18,600 samples (61 seconds) will be replaced by 4 segments containing 9,000 samples and labelled as AF. Finally, if the length is less than 9,000, we concatenate the recording with the same label to reach 9,000 samples. After pre-processing, the number of recordings for AF, Normal, Noisy and Other are 903, 5,959, 299, and 2,990, respectively, increasing from 8,528 to 10,151 in total.

ALGORITHM 1
PSEUDOCODE OF DATA LENGTH NORMALIZATION

```
IF the length of the recording is greater than 9,000 samples
    chop recording into 9,000 samples with 50% overlap
    between segments
IF the length of the recording is less than 9,000 samples
    DATA:=copy the recording
    append DATA in the back of the recording
    DO step 5 until the appended recording reaches 9,000
    samples
IF the length of the recording is equal to 9,000 samples
    preserve the recording
```

C. 1-D CNN Design

1) CNN Architecture

The design of deep neural network architectures is rather challenging, and it involves choices of several aspects such as performance metric, loss function, optimization algorithm, and

hyperparameter setting [32]. The hyperparameter setting includes the settings of the number of hidden layers, number of neurons and number of channels for every layer, learning rate, batch size, batch normalization, pooling and so forth. Transfer learning has been widely used in 2-D images. Several excellent pre-trained 2-D networks such as LeNet, AlexNet, VGG, Inception, and ResNet can be applied to other image processing tasks just with simple fine-tuning [33],[31]. However, there are very few pre-trained networks in one dimension, hence we design the network from scratch. Based on our experiences and utilizing the grid search [32], we obtain a 10-layer architecture through the process of trial and error.

The proposed architecture contains 10 convolutional blocks, 2 fully-connected layers, and a Softmax layer as output prediction, as shown in Table I. A convolutional block consists of a convolutional layer, a ReLU layer, and a Maxpooling layer. We add a Batch Normalization (BN) layer [30] after ReLU activation only in the first convolutional block to normalize the input layer by adjusting and scaling the activations. BN normalizes the input to zero mean and unit variance, which improves the performance and stability of deep neural network [32]. Five convolutional blocks are then followed with a dropout layer. The next convolutional blocks share the same structure and then apply a dropout layer. In the last block, we only apply a convolutional layer and a ReLU layer. The arrangements of batch normalization and pooling methods affect the prediction accuracy, which will be discussed in the following section.

As shown in Table I, the filter size of the convolutional layer starts with 32, meaning 32 kernels will convolve with input data and output 32 feature maps. The filter size is increased by multiplication of 2 every 2 blocks in order to retrieve more combinatory features. The kernel size in all layers is set to 5 for the reduction of computational load.

We downsample each convolutional output by means of pooling layer with a kernel size of 2 and use Softmax function in the last layer to produce prediction probability over the 4 output classes. Softmax function, defined in (1), maps the real-valued input into prediction probability. Ground truth labels are transformed into one-hot encoding vector. In our case, we have 4 labels: AF, Normal, Noisy and Other. Each label is a vector, containing either 1 or 0. If the label is "AF," the one-hot encoding vector will be [1, 0, 0, 0], giving the probability of each label. We set 1 to the first place because we want the predicted probability output to be as close as 1.

$$Softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1, \dots K \qquad (1)$$

TABLE I
PARAMETERS OF EACH LAYER OF PROPOSED 1-D CNN

Layers	Parameters	Activation
Conv1D	Filter 32 / kernel 5	ReLU
BN		
Maxpooling	2	
Conv1D	Filter 32 / kernel 5	ReLU
Maxpooling	2	
Conv1D	Filter 64 / kernel 5	ReLU
Maxpooling	2	
Conv1D	Filter 64 / kernel 5	ReLU
Maxpooling	2	
Conv1D	Filter 128 / kernel 5	ReLU
Maxpooling	2	
Conv1D	Filter 128 / kernel 5	ReLU
Maxpooling	2	
Dropout	0.5	
Conv1D	Filter 256 / kernel 5	ReLU
Maxpooling	2	
Conv1D	Filter 256 / kernel 5	ReLU
Maxpooling	2	
Dropout	0.5	
Conv1D	Filter 512 / kernel 5	ReLU
Maxpooling	2	
Dropout	0.5	
Conv1D	Filter 512 / kernel 5	ReLU
Flatten		
Dense	128	ReLU
Dropout	0.5	
Dense	32	ReLU
Dense	4	Softmax

2) CNN Learning

The CNN model is obtained by training with the stochastic gradient descent (SGD) optimization algorithm [32]. The SGD utilizes the backpropagation (BP) method to learn the parameter set of the neural network by minimizing the loss function which corresponds to the difference between network prediction and the ground truth.

In this CNN, we choose the cross-entropy as loss function, $J(\boldsymbol{x}, \boldsymbol{y}, \theta)$ defined in Eq.(2). The learning of the network is to minimize $J(\boldsymbol{x}, \boldsymbol{y}, \theta)$ with respect to network parameter set θ.

$$J(\boldsymbol{x}, \boldsymbol{y}, \theta) = -\sum_{i=1}^{K} x_i \log y_i, \qquad (2)$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the ground truth and the predicted output of the CNN, respectively, and K is the minibatch size.

In backpropagation training, a variant of SGD, adaptive moment estimation (Adam) [34], is utilized to update the θ iteratively by [32]

$$\theta_t \leftarrow \theta_{t-1} - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}, \quad (3)$$

where η is a fixed learning rate; ε is a very small constant; $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first moment estimate and the biased-corrected second moment estimate, which are defined in [34].

Adam is an adaptive learning rate method, which computes individual learning rates for different parameters. The default learning rate is 0.001 [34]. Generally, a large learning rate will make model learn faster while small learning rate will maintain better stability in learning process. In the 1-D CNN, dropout layer is also employed between the two convolutional blocks to avoid overfitting. Dropout layer will randomly choose a percentage of neurons and only update those weights without chosen during training time but will not be used during inference time [32]. We set dropout parameter to be 0.5, which means half of the neurons will not be updated.

III. NUMERICAL ANALYSIS

A. Dataset

As mentioned before, the dataset we used contains four classes: AF, Normal, Noisy and Other. Figure 4 [36] demonstrates the examples of ECG waveforms of the four classes. The "Other" represent that the recording can be clearly classified but does not belong to any of AF, Normal or Noisy. As mentioned in the previous section, the original dataset contains 8,528 recordings. After data length normalization, the number of recordings becomes to 10,151. The increase of data samples is beneficial for training in avoidance of overfitting.

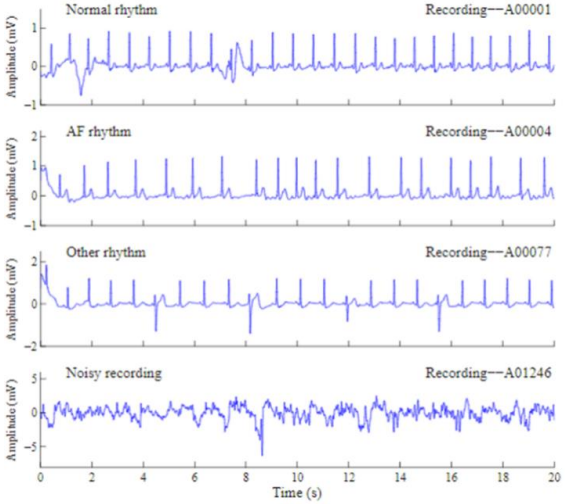

Fig. 4. ECG examples of four classes: Normal, AF, Other and Noisy.

B. Evaluation Metrics

As stated before, the proposed system predicts the input ECG record as one of the four classes. We adopt F_1 score to evaluate the prediction (detection) performance, which is calculated as

$$F_1(\%) = \left(\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}\right) * 100, \quad (4)$$

Precision and recall metrics, can be derived from the confusion matrix, which shows the performance of a classification algorithm. Each row of the matrix stands for predicted class while each column represents ground truth class. The subscript indicates the number of classifications. For example, $A_{\sim}$ represents AF as ground truth but predicted noisy as the outcome. A confusion matrix of 4 classes [36] is shown in Table II.

TABLE II
CONFUSION MATRIX OF 4 CLASSES

		Ground truth			
		AF (A)	Normal (N)	Noisy (~)	Other (O)
Predicted	AF(a)	A_a	N_a	$\sim_a$	O_a
	Normal (n)	A_n	N_n	$\sim_n$	O_n
	Noisy (~)	$A_{\sim}$	$N_{\sim}$	$\sim_{\sim}$	$O_{\sim}$
	Other (o)	A_o	N_o	$\sim_o$	O_o

To calculate F_1 of each class, we transform confusion matrix from 4 by 4 to 2 by 2. Each row and column contain 2 classes: target label and remaining label. A 2 by 2 confusion matrix of AF is shown in Table III. "Non-AF" includes Normal, Noisy, and Other classes.

TABLE III
CONFUSION MATRIX OF 2 CLASSES

		Ground truth	
		AF(A)	Non-AF (NA)
Predicted	AF(a)	A_a	NA_a
	Non-AF (na)	A_{na}	NA_{na}

Precision, recall, and accuracy [35],[37] of AF are calculated as follows:

$$\text{Precision} = \frac{A_a}{A_a+NA_a}, \quad (5)$$

$$\text{Recall} = \frac{A_a}{A_a+A_{na}}, \quad (6)$$

$$\text{Accuracy} = \frac{A_a+NA_{na}}{A_a+NA_a+A_{na}+NA_{na}}. \quad (7)$$

Accuracy is used in most cases in terms of evaluating how good the classification is. However, accuracy is inadequate for unbalanced dataset due to the majority of negative class. Therefore, we use F_1 score as an alternative measure since it is

a balance of precision and recall metrics and is useful when dealing with uneven class distribution [35].

F_1 scores for the other classes, i.e. Normal (F_{1N}), Noise ($F_{1\sim}$) and Other (F_{1O}), are calculated in the same way as AF (F_{1A}). The overall detection performance of proposed method is evaluated by the average of F_1 scores of the 4 classes [36] as

$$\text{Average } F_1 = \frac{F_{1N}+F_{1\sim}+F_{1A}+F_{1O}}{4}. \quad (8)$$

C. K-fold Cross-validation

The goal of cross-validation (CV) is to evaluate the generalization ability of a trained model [32],[38],[39],[40]. K-fold CV is the most popular method in various applications of machine learning. Stratified K-fold CV is a stratified-sampling version of K-fold which returns stratified folds: each set contains approximately the same percentage of samples of each target class as the complete set. Stratified K-fold CV is superior to regular K-fold CV, especially for imbalanced classifications.

As mentioned in previous section, the sample distribution of the four classes in the pre-processed dataset which contains 10,151 segments (AF=903, Normal=5959, Noisy=299 and Other=2,990) is imbalanced. Therefore, we utilize stratified K-fold CV rather than regular K-fold CV.

In order to find the best K value, we divide the pre-processed dataset into training and test subsets with different size ratios, and then do training and testing for each size ratio. The results are shown in Table IV. It indicates that the train/test ratio of 80:20 achieves the best detection performance, which corresponds to the K=5. In addition, the training time for 5-fold CV is 2,085 seconds (34.75 minutes), and inference time for 2031 recordings is 1.0033 seconds.

TABLE IV
EXAMPLE OF F_1 SCORE OF EACH FOLD FOR EVERY CLASS

Train/test	AF	Normal	Noisy	Other	Average
60:40	72.0	90.0	60.0	69.0	72.75
70:30	76.0	90.0	60.0	70.0	74.00
80:20	77.0	89.0	67.0	74.0	76.75
90:10	76.9	90.0	62.1	72.8	75.45
5-fold	79.0	91.0	65.0	76.0	77.75

TABLE V
EXAMPLE OF F_1 SCORE OF EACH FOLD FOR EVERY CLASS

F_1 Score	AF	Normal	Noisy	Other	Mean	σ
Fold_1	80.7	91.8	58.7	77.3	77.1	11.9
Fold_2	75.3	90.0	66.1	74.1	76.4	8.6
Fold_3	78.0	90.7	63.4	78.0	77.5	9.7
Fold_4	80.6	90.7	72.9	76.4	80.2	6.7
Fold_5	81.1	90.3	65.2	74.4	77.8	9.2
FAVG_ F_1	79.1	90.7	65.3	76.0		

Table V shows an example of F_1 score of each fold for every class. The average F_1 score (abbreviated as FAVG_F_1) of a class is calculated by averaging F_1 of all folds. We use average score of each class since it can avoid selection bias problem. The result shows that the FAVG_F_1 of AF, Normal, Noisy, and Other are 79.1%, 90.7%, 65.3%, and 76%, respectively. The table also lists the mean and standard deviation (σ) over all classes for each fold.

D. Hyperparameter Optimization

We perform grid search algorithm over number of layers, kernel size, batch size, and learning rate, for hyperparameter optimization in training. Given a set of parameters for training, grid search yields an optimal model which maximizes accuracy on independent datasets. The suggested learning rate of 0.001 in Adam optimizer [34] is used as a base. In order to find the best accuracy around the base learning rate, we choose one parameter greater than the base and three more less than the base to form a set for learning rate as Lr ∈ {0.00005, 0.0001, 0.0005, 0.001, 0.005}. We select 5 elements to form a set which starts from 30 and is increased by 20; i.e., Bs ∈ {30, 50, 70, 90, 110}. Three kinds of kernel size are selected as Ks ∈ {3, 5, 7}. As for number of layers in our case, the upper limit is 12 since we decrease the dimensionality of feature map after every convolutional layer. We define the set of number of layers as N ∈ {8, 9, 10, 11}. The combination of the above parameter sets yields 300 neural network architectures. We apply the grid search to obtain an optimal architecture and use 4 GPUs (Nvidia GTX 1080 Ti) to speed up training.

E. Results and Analysis

1) Prediction Accuracy

Each recording is assigned a label AF, Normal, Noisy or Other. The proposed 1-D CNN model will infer label on the test set. Using the grid search, we can obtain the best model and evaluate the average F_1 score, which is the average of the FAVG_F_1 of the four classes.

Applying the grid search to the dataset, we obtain the results demonstrated in Table VI. The table shows the best average F_1 score under various combinations of network hyperparameters including number of convolutional layers, filter kernel size, batch size and learning rate. For instance, for the CNN with 8 convolutional layers, the first row of the table lists the best F_1 scores of kernel sizes of 3, 5, 7 with corresponding optimal batch sizes of 50, 50, 70 and learning rates of 0.001, 0.001, 0.0005, respectively. It is noted that for 11 convolutional layers, there is no output when kernel size is greater than 3. This is because no sufficient data to perform convolution in the last convolutional layer since the feature maps are downsampled using pooling every convolutional layer. This table also shows the total number of network parameters for each architecture combination.

TABLE VI
THE BEST AVERAGE F_1 UNDER VARIOUS COMBINATIONS OF HYPERPARAMETERS

Layer	Kernel size	Batch size	Learning rate	Average F_1	σ	Total number of parameters
8	3	50	0.001	69.4	12.5	2,558,340
	5	50	0.001	72.2	12.9	2,687,428
	7	70	0.0005	73.6	11.4	2,816,516
9	3	50	0.001	76.6	9.6	1,608,324
	5	50	0.0001	77.2	9.9	1,868,484
	7	30	0.0005	76.8	11.4	2,128,644
10	3	90	0.0005	77.0	9.4	2,428,292
	5	30	0.0001	77.8	9.1	3,212,740
	7	50	0.001	76.4	10.6	3,997,188
11	3	50	0.0005	77.1	9.9	2,625,412
	5	N/A	N/A	N/A	N/A	N/A
	7	N/A	N/A	N/A	N/A	N/A

Table VI indicates the best combination is: number of layers=10, kernel size=5, learning rate= 0.0001, batch size =30. The combination reaches average F_1 score of 77.8%. Since the batch size of 30 is the searching lower bound, we further run a small grid search with batch sizes of 10 and 20, and the result is displayed in Table VII. It proves batch size of 30 is the best parameter.

TABLE VII
AVERAGE F_1 SCORE UNDER DIFFERENT BATCH SIZES

Number of layers	Kernel size	Batch size	Learning rate	Average F_1	Total number of parameters
10	5	10	0.0001	76.4	3,212,740
		20		77.1	3,212,740
		30		77.8	3,212,740

Figure 5 shows the confusion matrix of the first fold in Table V. The diagonal shows the number of correctly predicted records for each class. The off-diagonals display the number of misclassifications for each class. It is seen that 21 AF records are misclassified as Other. This is because some AF records contain characteristics of class "Other." An example of AF record which is misclassified as Other is shown in Figure 6. In this figure, the orange rectangle shows that the ECG waveform has AF characteristics, which is affected by the abnormal mode of the R-R interval, while the yellow rectangle indicates that the ECG waveform does not have the characteristics of AF, Noisy, or Normal.

Figure 5 also indicates imbalanced dataset problem. For the test set, the total number of records of Normal class is 1215, and the total number of the records of the remaining classes is only 816. In other words, Normal class dominates the training, which makes the learned model have the possibility of tilting to the normal class.

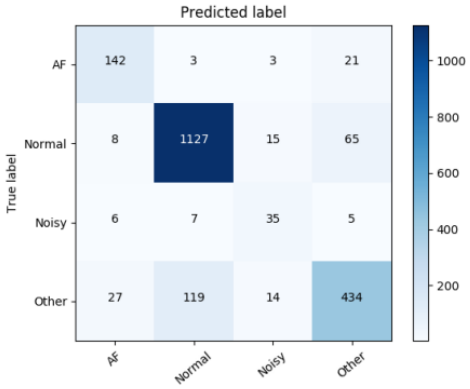

Fig. 5. Confusion matrix of the first fold in Table V

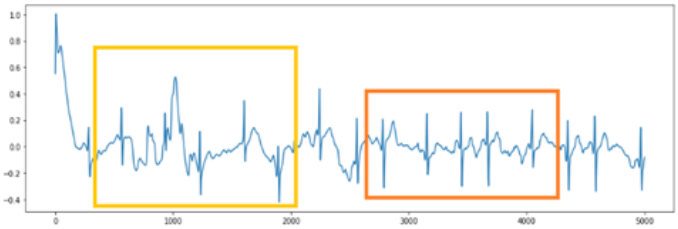

Fig. 6. AF record which is misclassified as Other

2) Network Architecture Analysis

Batch Normalization is a renowned regularization technique that accelerates training [30] and is widely used in modern network architectures [25]. Maxpooling is a strategy that downsamples feature maps and is mostly placed after convolutional layer for feature reduction [41]. Although these strategies have been proved to be robust and useful in several application domains, we conduct an experiment to look into the performance changes under various arrangements of Batch Normalization and Maxpooling. The proposed 1-D CNN architecture shown in Table I contains only one Batch Normalization layer placed after the first convolutional layer, and one Maxpooling layer for every convolutional layer except the last convolution layer. In addition, we replace Maxpooling layers of the 1-D CNN with average pooling. For comparison, we denote the two networks as Proposed-1 (Maxpooling) and Proposed-2 (Average pooling), respectively.

Table VIII shows the average scores of four labels for the two proposed networks and their variants. Here we train four variants of the Proposed-1 and one variant of the Proposed-2. The variants of Proposed-1 include (a) All-BN: add one Batch Normalization layer after every convolutional layer, (b) NO-BN: No Batch Normalization layer in the network, (c) Maxpooling: add the Proposed-1 with one Maxpooling layer before the Flatten layer, (d) Max-Average pooling: add the Proposed-1 with one Average pooling layer before the Flatten layer. The variant of Proposed-2 is to add one average pooling layer before Flatten layer into the network, which is denoted as Average pooling in Table VIII.

The detection performance of All-BN degrades significantly, as compared to NO-BN or our Proposed-1 architecture. This is due to gradient explosion [42] in deep layer that make network hard to train. Furthermore, Batch Normalization leads to instability of training [42]. Training/validation accuracy of the models All-BN and NO-BN are depicted in Figure 7 and 10, respectively. It is seen that the accuracy curve on validation set fluctuate drastically for ALL-BN (Figure 7) while the accuracy curve for NO-BN (Figure 8) is relatively smooth.

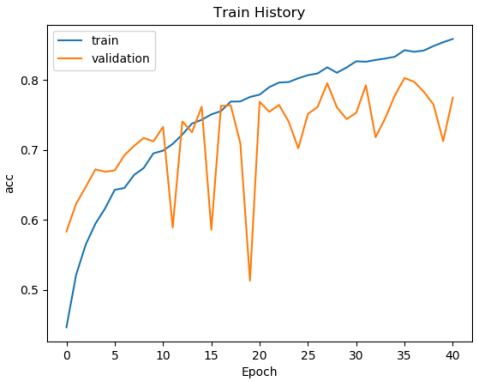

Fig. 7. Training accuracy of All-BN

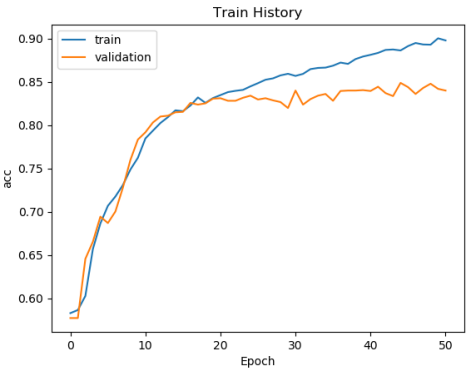

Fig. 8. Training accuracy of NO-BN

A vanilla CNN usually combines convolutional layer and Maxpooling layer as a block and repeat that before Flatten [41]. We investigate how Maxpooling layer that is placed before Flatten layer affects accuracy. The result shown as "Maxpooling" in Table VIII reflects that the accuracy degradates significantly, and this variant yields the worst accuracy. The reason can be traced back to the characteristics of Maxpooling. In Maxpooling operation, it only keeps the maximum element and remove the others. This method may not preserve the originality of data and is likely to eliminate the distinguishing feature in the same pooling region [43].

In order to investigate the effect of pooling methods, we replace Maxpooling in Proposed-1 with Average pooling, and the result is shown as Proposed-2 in Table VIII. It is obvious that Proposed-2 outperforms Proposed-1 and its variants. This may be due to the effect that Average pooling keeps most information from previous layer and is able to pass down layer by layer. In addition, the performance of Proposed-2 slightly degrades if we add one Average pooling layer before Flatten layer. But in Proposed-1, the F_1 score drop by 10% if we add one Maxpooling layer before Flatten layer. The experiment shows that using Average pooling is more stable than using Maxpooling.

TABLE VIII
TRAINING RESULTS OF PROPOSED ARCHITECTURES AND THEIR VARIANTS

Neural Networks	Average F_1 score	Total number of parameters
Proposed-1	77.8	3,212,740
All-BN	69.2	3,216,644
No-BN	76.2	3,212,676
Maxpooling	67.7	2,885,060
Max-Average pooling	75.6	2,885,060
Proposed-2	78.2	3,212,740
Average pooling	77.4	2,885,060

3) Network Complexity Analysis

To evaluate the network complexity, we estimate the total number of network training parameters using deep learning platform Keras. The parameters of each layer, excluding non-trainable layers such as pooling layer, dropout, and Flatten layer, are shown in Table IX. The total number of network training parameters of the proposed 1-D CNN is approximately 3 million.

For comparison, we also build up the network CRNN using Keras and calculate the total number of network training parameters of the CRNN. The CRNN architecture mainly consists of two networks: 2-D CNN and LSTM. The 2-D CNN uses four convolutional blocks. Each block is made up of six 2-D convolutional layers, followed by batch normalization and ReLU activation. The last layer of a block applies Maxpooling with kernel size of 2. Feature maps from the Maxpooling layer are flattened before feeding into a 3-layer bidirectional LSTM network. The total number of training parameters of CRNN is 10,149,440, which is three times more than ours and requires huge computation in training. Note that the transformation of 1-D ECG signal to 2-D spectrogram is not taken into account. In addition, 2-D convolution of 24 layers and 3-layer LSTM dominate the network complexity of the CRNN.

4) Comparison of Various Methods

To prove the effectiveness of our AF detection system, we compare it with the existing DL-based methods [24], [26], [27]. The DL-based methods include 1-D schemes: ResNet-1, ResNet-2, and CL3-I (CL3 experiment I [26]), and 2-D scheme: CRNN, as mentioned in Section I. The comparison metrics include detection accuracy of cross-validation and total number of network training parameters.

A detection accuracy comparison of our model with the existing DL-based methods for each class is summarized in Table X. It indicates that our proposed method achieves better prediction accuracy for all classes. Besides, the detection performance of AF of this method is more than 4% higher than the second best CRNN.

Table XI shows the comparison using the metrics of Average F_1 score of 4-classes and Average F_1 score of 3-classes (excluding Noisy), and the total number of parameters. The results indicate that the proposed networks achieve average F_1

score higher than the existing networks. The existing 1-D schemes, ResNet-1, ResNet-2 and CL3-I, perform much worse than our networks and CRNN in Average F_1 score of 4-classes. In addition, our network outperforms the CRNN with much lower network complexity (less than 1/3). This implies that both training cost and implementation cost of the proposed deep neural network are significantly lower than those of CRNN.

TABLE IX
TRAINING PARAMETERS OF THE PROPOSED 1-D CNN

Layer type	Output Shape	Param
Conv1D	8996 x 32	192
Batch Normalization	8996 x 32	128
Conv1D	4494 x 32	5152
Conv1D	2243 x 64	10304
Conv1D	1117 x 64	20544
Conv1D	554 x 128	41088
Conv1D	273 x 128	82048
Conv1D	132x 256	164096
Conv1D	62 x 256	327936
Conv1D	27 x 512	655872
Conv1D	9 x 512	1311232
Dense	128	589952
Dense	32	4128
Dense	4	132
Total number of network training parameters: 3,212,740		

TABLE X
COMPARISON OF PREDICTION ACCURACY OF VARIOUS METHODS

Methods	AF (A)	Normal (N)	Noisy (~)	Other (O)
Proposed-1	79.1	90.7	65.3	76.0
Proposed-2	80.8	90.4	66.2	75.3
CRNN	76.4	88.8	64.5	72.6
ResNet-1	65.7	90.2	64.0	69.8
ResNet-2	67.7	88.5	65.6	66.6
CL3-I	76.0	90.1	47.1	75.2

TABLE XI
COMPARISON OF AVERAGE F_1 SCORE AND TOTAL NUMBER OF PARAMETERS OF VARIOUS METHODS

Methods	Average F_1 score of A, N, ~, and O	Average F_1 score of A, N, and O	Total number of parameters
Proposed-1	77.8	81.9	3,212,740
Proposed-2	78.2	82.2	3,212,740
CRNN	75.6	79.3	10,149,440
ResNet-1	72.4	75.2	10,466,148
ResNet-2	72.1	74.3	1,219,508
CL3-I	72.1	80.4	206,334

IV. CONCLUSION

This paper has developed an end-to-end 1-D CNN for the AF detection from time-series ECG data. By studying the impact of the batch normalization and pooling methods on detection accuracy and combing the grid search method to obtain optimal hyperparameters, we designed a simple, yet, effective 1-D CNN network. We also developed a length normalization algorithm to make a variable-length ECG record becoming a fixed-length one, which solves variable-length problem and augment more training data. Through exhaustive evaluation, we prove our method achieves better cross-validation detection accuracy than the existing methods. In addition, the proposed method reduces network complexity significantly, as compared with the second-ranked method CRNN [27]. By applying spatial pyramid pooling [44], which does not require fixed input size, in replacement of Flatten layer, and investigating a better way to solve the problem of extremely imbalanced dataset will be the future directions.

REFERENCES

[1] "Atrial Fibrillation," Available: https://www.mayoclinic.org/diseases-conditions/atrial-fibrillation/symptoms-causes/syc-20350624, Last accessed date: Oct 18, 2019.

[2] Nathalie Conrad, Andrew Judge, Jenny Tran, Hamid Mohseni, Deborah Hedgecott, Abel Perez Crespillo, Moira Allison, Harry Hemingway, John G Cleland, John J V McMurray, and Kazem Rahimi, "Temporal trends and patterns in heart failure incidence: a population-based study of 4 million individuals," *Lancet* 391 (10120): Nov. 2017.

[3] Christopher Cook, Graham Cole, Perviz Asaria, Richard Jabbour, and Darrel P. Francis, "The annual global economic burden of heart failure," *International Journal of Cardiology*, vol. 171, no. 3, pp. 368 - 376, 2014.

[4] Sunil T. Mathew, Jigar Patel, and Satheesh Joseph, "Atrial fibrillation: mechanistic insights and treatment options," *European Journal of Internal Medicine*, vol.20, no. 7, pp. 672–681, 2009.

[5] Xiaolin Zhou, Hongxia Ding, Benjamin Ung, Emma Pickwell-MacPherson, and Yuanting Zhang, "Automatic online detection of atrial fibrillation based on symbolic dynamics and Shannon entropy," *BioMedical Engineering OnLine*, 13:18, 2014

[6] Vsr Kumari, and P. Rajesh Kumar, "Cardiac arrhythmia prediction using improved multilayer perceptron neural network," *Int. J. Electron., Commun., Instrum. Eng. Res. Develop*, vol. 3, no. 4, pp. 73–80, 2013.

[7] Gari D Clifford, Chengyu Liu, Benjamin Moody, Li-wei H. Lehman, Ikaro Silva, Qiao Li, A E Johnson, and Roger G. Mark, "AF classification from a short single lead ECG recording: the PhysioNet Computing in Cardiology Challenge 2017," *2017 Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4.

[8] Xionghui Tang and Lihao Shu, "Classification of electrocardiogram signals with RS and quantum neural networks," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 2, pp. 363-372, 2014.

[9] Stanislaw Osowski and Tran Hoai Linh, "ECG beat recognition using fuzzy hybrid neural network," *IEEE Trans. Biomed. Eng.*, vol. 48, no. 11, pp. 1265–1271, 2001.

[10] Tran Hoai Linh, Stanisław Osowski, and Maciej Stodolski, "On-line heartbeat recognition using Hermite polynomials and neuron-fuzzy network," *IEEE Trans. Instrum. Meas*., vol. 52, no. 4, pp. 1224–1231, 2003.

[11] Sucharita Mitra, Madhuchhanda Mitra, and B. B. Chaudhuri, "A rough set-based inference engine for ECG classification," *IEEE Trans. Instrum. Meas*., vol. 55, no. 6, pp. 2198–2206, 2006.

[12] Farid Melgani and Yakoub Bazi, "Classification of electrocardiogram signals with support vector machines and particle swarm optimization," *IEEE Trans. Inf. Technol. Biomedicine*, vol. 12, no. 5, pp. 667-677, 2008.

[13] Mariano Llamedo and Juan Pablo Martínez, "Heartbeat classification using feature selection driven by database generalization criteria," *IEEE Trans. Biomedical Eng.*, vol. 58, no. 3, 2011.

[14] Mohammad R. Homaeinezhad, Seyyed Abbas Atyabi, E. Tavakkoli, Hamid Najjaran Toosi, Ali Ghaffari, and Reza Ebrahimpour, "ECG arrhythmia recognition via a neuro-SVM–KNN hybrid classifier with virtual QRS image-based geometrical features," *Expert Syst. Appl.*, vol. 39, pp. 2047–2058, 2012.

[15] Hari Mohan Rai, Anurag Trivedi, and Shailja Shukla, "ECG signal processing for abnormalities detection using multi-resolution wavelet transform and artificial neural network classifier," *Measurement*, vol. 46, pp. 3238–3246, 2013.

[16] Yun-Chi Yeh, Wen-June Wang, and Che Wun Chiou, "A novel fuzzy c-means method for classifying heartbeat cases from ECG signals," *Measurement*, vol. 43, pp. 1542–1555, 2010.

[17] Manal M. Tantawi, Kenneth Revett, Abdel-Badeeh M. Salem, and Mohamed F. Tolba, "Fiducial feature reduction analysis for electrocardiogram (ECG) based biometric recognition," *Journal of Intelligent Information Systems*, vol. 40, pp. 17–39, 2013.

[18] A. De Gaetano, S. Panunzi, F. Rinaldi, A. Risi, and M. Sciandrone, "A patient adaptable ECG beat classifier based on neural networks," *Applied Mathematics and Computation*, vol. 213, pp. 243–249, 2009.

[19] Shreyasi Datta, Chetanya Puri, Ayan Mukherjee, Rohan Banerjee, Anirban Dutta Choudhury Rituraj Singh, Arijit Ukil, Soma Bandyopadhyay, Arpan Pal, and Sundeep Khandelwal, "Identifying normal, AF and other abnormal ECG rhythms using a cascaded binary classifier," 2017 *Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4.

[20] Vykintas Maknickas and Algirdas Maknickas. "Atrial fibrillation classification using QRS complex features and LSTM," *2017 Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4. doi: 10.22489/CinC.2017.350-114

[21] Nadi Sadr, Madhuka Jayawardhana, Thuy T Pham, Rui Tang, Asghar Tabatabaei Balaei, and Philip de Chazal, "A low-complexity algorithm for detection of atrial fibrillation using an ECG," *Physiological Measurement* vol. 39, no. 6, May 2018.

[22] B. S. Chandra, C. S. Sastry, S. Jana, and S. Patidar, "Atrial fibrillation detection using convolutional neural networks," *2017 Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4.

[23] Kan Luo, Jianqing Li, Zhigang Wang, and Alfred Cuschieri, "Patient-specific deep architectural model for ECG classification," *Journal of Healthcare Engineering*, vol. 2017, Article ID 4108720, 13 pages, 2017.

[24] Fernando Andreotti, Oliver Carr, Marco A. F. Pimentel, Adam Mahdi, and Maarten De Vos, "Comparing feature-based classifiers and convolutional neural networks to detect arrhythmia from short segments of ECG," *2017 Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4.

[25] Awni Y. Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H. Tison, Codie Bourn, Mintu P. Turakhia, and Andrew Y. Ng, "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks," *Nature Medicine* volume 25, pages65–69 (2019)

[26] Philip Warrick, and Masun Nabhan Homsi, "Cardiac arrhythmia detection from ECG combining convolutional and long short-term memory networks," *2017 Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4.

[27] Martin Zihlmann, Dmytro Perekrestenko, and Michael Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," *2017 Computing in Cardiology (CinC)*, vol. 44, Rennes, 2017, pp. 1-4.

[28] C. H. Hsieh, J.Y. Chen, and B. H. Nien, "Deep learning-based indoor localization using received signal strength and channel state information," *IEEE Access*, vol.7, 2019, pp.33256-33267.

[29] Zheng Zhou, Youzuo Lin, Zhongping Zhang, Yue Wu, and Paul Johnson, "Earthquake detection in 1-D time series data with feature selection and dictionary learning," *Seismological Research Letters* (2019) 90 (2A): 563-572.

[30] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. 37, pp. 448-456, July 2015.

[31] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu, "A Survey on Deep Transfer Learning," 27th *International Conference on Artificial Neural Networks*, Rhodes, Greece, October 4–7, 2018, Proceedings, Part III. 10.1007/978-3-030-01424-7_27.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, MIT Press, 2016.

[33] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang, "A survey of transfer learning," *Journal of Big Data* 3:9, 2016.

[34] Diederik P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2014.

[35] Yutaka Sasaki, "The truth of the f-measure," *Teaching, Tutorial materials*, Version: 26th October 2007.

[36] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003).

[37] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, Keying E. Ye, Probability and Statistics for Engineers and Scientists, ninth edition, *Pearson*.

[38] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning, Springer, 2008.

[39] Gareth James, Daniela Witten, Trevor Hastie, and Rob Tibshirani, An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) 1st ed. 2013, Corr. 7th printing 2017 Edition.

[40] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[41] Karen Simonyan, and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *2019 International Conference on Learning Representations.*

[42] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz, "A mean field theory of batch normalization," *2019 International Conference on Learning Representations*.

[43] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei, "Mixed pooling for convolutional neural networks," *Rough Sets and Knowledge Technology: 9th International Conference*, pp. 364-375.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 37. 10.1109/TPAMI.2015.2389824.