Author: Hermela Shimelis, PhD, MS-DS

Classification of arrhythmias using ECG

Summary:

The aim of this project is to classify arrhythmias into four categories: atrial fibrillation (AFIB), grouped supraventricular tachycardia (GSVT), sinus bradycardia (SB), and sinus rhythm (SR) using publicly available ECG data from 10,646 patients, by employing different model architectures, including 1D convolutional neural network (CNN), Long Short-Term Memory Networks (LSTMs), ResNet and InceptionTime. For the given dataset, the InceptionTime model showed the best performance on classifying the four arrhythmias.

Introduction and Literature review

Heart diseases affect millions of people every year. Arrhythmias are one of the most common and serious types of heart conditions, characterized by abnormal heart rhythms, including fast, slow, irregular, or uncoordinated heartbeats. Electrocardiogram (ECG) is a non-invasive diagnostic test that captures the heart's electrical activity, providing information about the heart's condition. It is primarily used for diagnosing arrhythmias, ischemic heart disease, and heart attacks (*1*). ECG records the heart's electrical activity as waveforms, which consist of P waves, QRS complexes, and T waves (Figure 1).

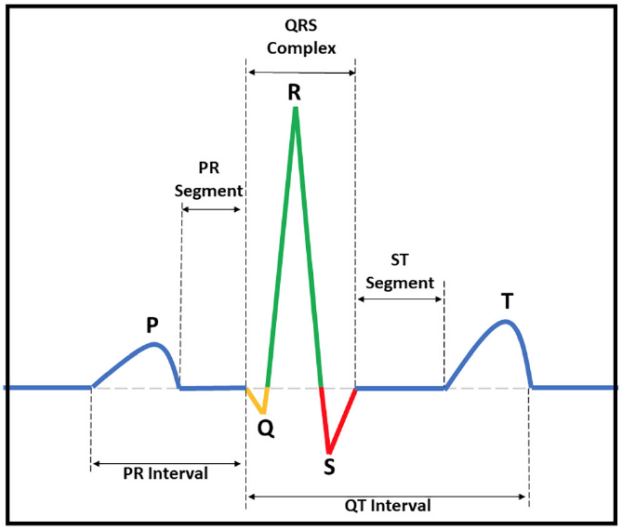


Figure 1. Waveforms showing components of ECG, showing atrial depolarization (P-wave), ventricular depolarization (QRS complex wave), and repolarization (T-wave).

Physicians analyze these waveforms to make a diagnosis. However, this manual approach has several limitations: the accuracy is highly dependent on the physician's experience, it is labor-intensive, and it is susceptible to noise and artifacts (*1*). Therefore, automating the diagnosis of arrhythmias and other heart conditions using ECG could significantly enhance the accuracy of detecting heart conditions.

Several machine learning and deep learning models have been extensively used for rhythm classification from ECGs (*1*, *2*). This is partly due to the availability of multiple databases that made ECG data available for research (*1*). ECG apps have in fact been successfully deployed. The Apple Watch can classify arrhythmias by monitoring electrical heart activity using digital crown and back crystal to record ECG. This tool can classify ECG as sinus rhythm (normal rhythm) or atrial fibrillation, low or high heart rate. Deep learning models (DL) have shown several advantages over traditional machine learning models for ECG classification. For example, DL models have automatic feature extraction capabilities, eliminating the need for feature engineering, they are robust to noise, selected DL models have shown better ability in capturing temporal features that are important for detecting arrhythmias (*1*). The list of recent deep learning models that have shown superior performance have been extensively reviewed by Ansari et. al., (*1*).

Although numerous deep learning (DL) models have been proposed for arrhythmia classification, this project focuses on state-of-the-art DL models specifically suited for long time series classification. This selection is driven by the nature of the ECG data used, which contains 5,000 timestamps. While extensive literature exists on ECG classification, the majority of studies use raw waveforms, differing from the preprocessed data used in this report. Consequently, the results of these studies may not be directly comparable to the results of the current study. This review is therefore limited to studies comparable to the current project in terms of DL model architecture and data source. A few studies that have employed a 1D convolutional neural network, LSTM, a combination of CNN and LSTM, ResNet, or InceptionTime network are summarized below.

The 1D convolutional neural network was introduced for ECG classification due to its one-dimensional structure and efficiency in capturing long time series in a fast and accurate manner (*3*). After it was introduced, several studies have achieved high accuracy by employing 1D-CNN with various depths and structures (*4*, *5*). Yildirim et al. (*6*) employed a deep neural network model combining CNN and LSTM layers to classify arrhythmia in Chapman's ECG data, the same dataset used in this report. The model includes six 1D convolutional layers, one LSTM layer with MaxPooling, batch normalization, and dropout layers (*6*). The model achieved 96.13% overall accuracy on the four rhythm types that were evaluated in this report. Interestingly, the study found that the model performed well even

when using only one of the 12 leads. A broad review of studies on ECG arrhythmia classification has shown that combining different model structures yields superior performance. The combination of CNN and LSTM layers has demonstrated some of the highest F1-scores, as reported by Midani et al. (*7*) (97.5%), Chen et al. (*8*) (98.7%), and Tan et al. (*9*) (99.5%). While the 1D CNN and LSTM models have been used extensively, recent DL models such as ResNet and InceptionTime have also been implemented in ECG classification (*10*).

Strodthoff et al., (*10*) employed seven neural network models, providing benchmarks and insights from PTB-XL 12 lead ECG from 18,885 patients. The performance of these models on rhythm classification is summarized in the table below.

Table 1. Comparison of DL models performance on arrhythmia classification from ECG reported in Strodthoff et al., (*10*).

Model	**Maximum F1 score on arrhythmia classification**
LSTM	0.908
Bidirectional LSTM	0.907
Fully convolutional network	0.899
Resnet 1D	0.908
Inception 1D	0.917

This study provided good benchmarking results for DL time series classification algorithms for arrhythmia. As shown in the table, these models performed similarly, but the InceptionTime model showed the highest performance. This study has also served as a motivation for the current report to focus on one of the most popular and novel model architectures, the InceptionTime.

ECG signals are inherently sequential, capturing the heart's electrical activity over time. The temporal pattern is important for detecting normal and pathological status of the heart. For this reason, time series classification models are often used for arrhythmia classification with ECG. In the following section, model architectures of feature-based state-of-the- time series classification models are described briefly.

Time series classification algorithms

Several deep learning algorithms have been successfully used for time series classification problems, including Fully Convolutional Neural Network, RNN variant LSTM, ResNet, time LeNet, and InceptionTime. This report does not cover all of them, but they

have been described in detail by Fawaz et al., (*11*). For the current ECG classification, a combination of CNN-bidirectional LSTM, CNN-Gru, ResNet, and InceptionTime models have been used, and they are summarized here from the original review article. Each of these models have been briefly described in this paper. These architectures were selected due to their success in ECG and other time series classification, and compatibility with computational resources in hand. Given the following definition of multivariate time series (MTS):

MTS = $X = (x^1, \ldots, x^T)$ consists of *T* ordered elements $x^j \in \mathbb{R}^m$

Deep neural time series network is a composition of L parametric functions, where L is layers (one sample with time series) where each layer is considered a representation of input data (*11*). One layer, $l_i (i \in 1 \ldots L)\ or$ element of a time series contains neurons and takes as input the output of the previous layer ($l_{i\text{-}1}$), then the layer's output is calculated after non-linearity function is applied, which is controlled by a parameter $\theta_i\ or\ weight$. Given an input x, a neural network computes the following to predict the class.

$$f_L(\theta_L, x) = f_{L-1}(\theta_{L-1}, f_{L-2}(\theta_{L-2}, \ldots, f_1(\theta_1, x))) \quad (1)$$

f_i represents non-linearity function applied to layer l_i.

This process is essentially feed-forward propagation (*11*). In the time series network, given the input x, the output vector with an estimated probabilities of the input belonging to a given class is computed using the function f. The prediction loss is calculated using a cost function and weights are updated to minimize cost in backward propagation using gradient descent.

1D Convolutional Neural Network (CNN)

CNN architectures have been successfully used for ECG classification and other time series problems. For time series classification, convolution is like applying and sliding a filter over time series. The result of a convolution for a centered timestamp t is given by equation 2 (*12*).

$$C_t = f(\omega * X_{t-l/2:\, t+l/2} + b) \mid \forall\, t \in [1, T] \quad (2)$$

Where is the result of a convolution dot product applied on a univariate time series X of T with a filter ω of length l, a bias parameter b and nonlinear function f Rectified Linear Unit (ReLU). Applying several filters on a time series would allow learning multiple discriminative features (*12*). The same convolution, with the same filter values ω and b are used throughout the time stamps $t \in [1, T]$, which is referred to as weight sharing. Convolution is followed by an average or max pooling layer which takes an input time

series and reduces the length T by the length of the filer. Batch normalization is also included after the convolution layer to help the network converge quickly (*12*).

Recurrent Neural Network (RNN)

RNNs are specialized neural networks for recognizing patterns or sequences with direct connections that allow outputs from previous layers to be used as inputs, allowing the flow of information until the end of sequence. The final current state is used to calculate the final output. Long Short-Term Memory (LSTM) network is a special type of RNN capable of learning long-term dependencies. They were introduced to address the vanishing gradient problem that can occur with simple RNNs. LSTMs have a more complex architecture that includes memory cells and gates (input, output, and forget gates) to control the flow of information. This allows them to maintain and update information over longer periods, making them effective for time series prediction. LSTMs have been successfully used for arrhythmia classification using ECG (*13*).

InceptionTime

The InceptionTime model is deep learning architecture inspired by the Inception network used in computer vision, specifically designed to handle time series data. The original InceptionTime model proposed by Ismail et al., consists of an ensemble of five different inception networks, each initialized randomly (*14*). A single InceptionTime network contains two residual blocks, as opposed to ResNet, which includes three. For the inception network, each block contains three inception modules.

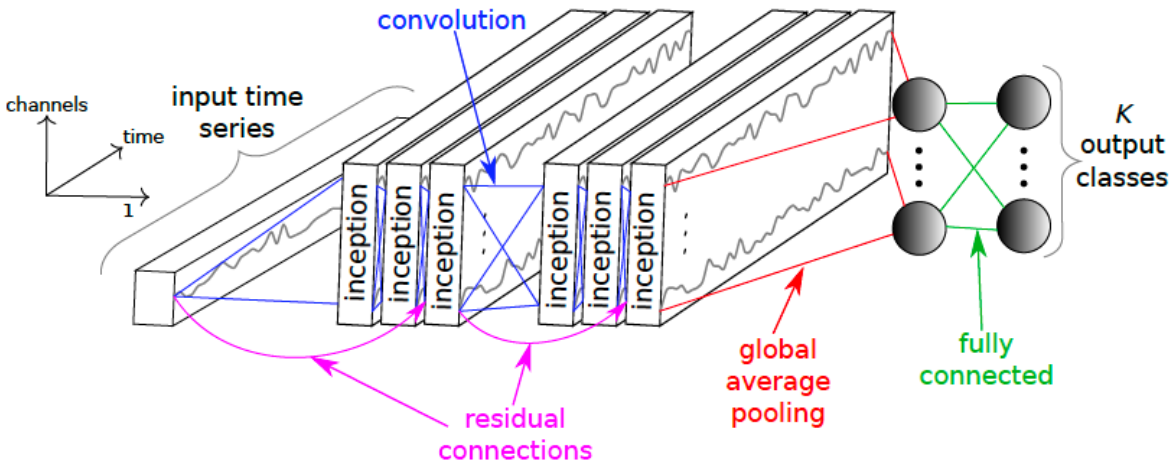


Figure 2. The InceptionTime network architecture. This figure is taken from (*14*) for illustration purposes.

The residual connection in an InceptionTime network uses a short connection to add the input of a layer to the output of subsequent Inception layer, effectively allowing the network to learn residual information, mitigating the vanishing gradient problem. After the residual blocks, a global average pooling layer averages the output of multivariate time series, which is fed into a fully connected softmax layer with the number of neurons equal to the number of classes.

Each inception module includes 1D convolutional layers with different filter sizes of length and a MaxPooling layer. The first component of the InceptionTime module is the bottleneck layer which uses 1D convolutions to reduce the number of input channels, essentially used for dimensionality reduction. This layer involves sliding m filters (e.g., 32 or 64) of length 1 with stride equal to 1. This reduces the dimensions of multivariate time series (MTS) with M dimensions (# features) to MTS with m dimensions (m feature maps), reducing dimensionality. The second component of the inception module involves sliding multiple filters of different lengths simultaneously on the same input time series fed from the bottleneck layer. In parallel, a MaxPooling layer is applied to reduce overfitting. Finally, the output of each independent parallel convolution and MaxPooling is concatenated to form the output of multivariate time series from the inception module (*14*) .

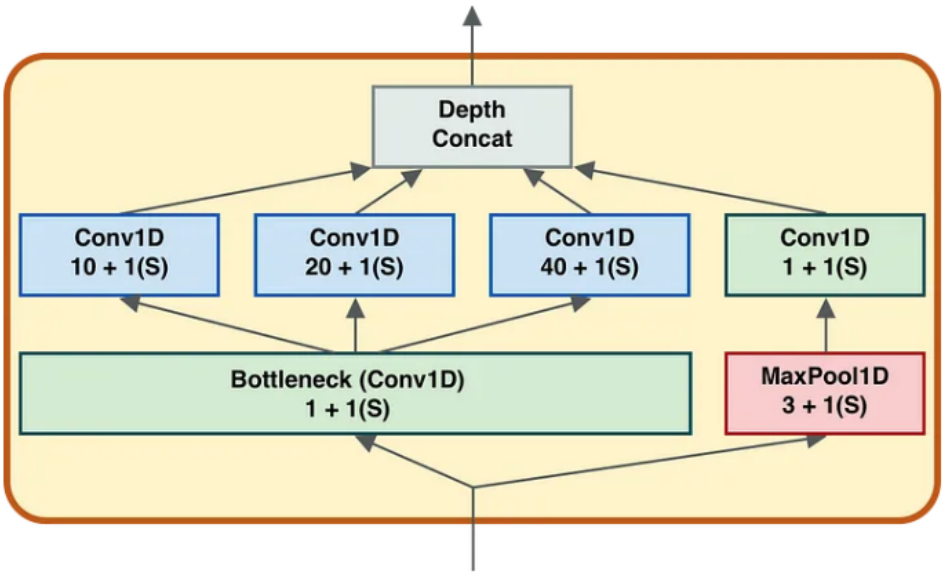


Figure 3. The inception module of InceptionTime. The first number in the boxes are the kernel size and the second number is the size of the stride.

Source: https://medium.com/towards-data-science/deep-learning-for-time-series-classification-inceptiontime-245703f422db

Multiple inception modules are stacked, and the weights (filter's values) are updated via backpropagation within each module. Equation (3) describes the ensembling of predictions by the IncepionTIme network

$$\hat{y}_{i,c} = \frac{1}{n}\sum_{j=1}^{n} \quad \sigma_c(x_{i,}, \theta_j) \mid \forall_c \in [1, C] \tag{3}$$

$y_{i,c}$
$= ensemble' output\ probability\ of\ having\ the\ input\ time\ series\ x_{i,} belonging\ to\ class\ c,$

$which\ is\ equal\ to\ the\ logistic\ output\ \sigma_c\ averaged\ over\ the\ n\ randomly\ initialized\ models.$

$$\theta_j\ is\ weight$$

For the final model, the default batch size of 64 was used. Ismail Fawaz et al., reported that batch size did not have much effect on accuracy under their experimental condition. However, the authors highlighted that filter length affects accuracy and longer time series may require longer filter size. The final representation of the input time series serves as input for the discriminative layer which is composed of a softmax. This layer outputs a probability distribution of the class variables.

Residual Network (ResNet)

The ResNet architecture is a deep neural network that includes 11 layers, 9 of which are convolutional followed by GlobalAveragePooling layer which averages the time series across the time dimension. The special characteristic of this ResNets is the linear residual connection between consecutive convolutional layers which links the output of a residual block to its input, effectively reducing the vanishing gradient problem (*12*). The network includes three residual blocks, followed by a GlobalAveragePooling layer and a final softmax classifier. Each residual block includes 3 convolutions (filter length set to 8, 5, 3) whose output is added to the residual blocks' input, which is fed to the next layer. The number of filters for all convolutions is fixed to 64 in the first block and 128 in next two blocks. After convolution, the output is subjected to batch normalization and ReLu activation (*15*).

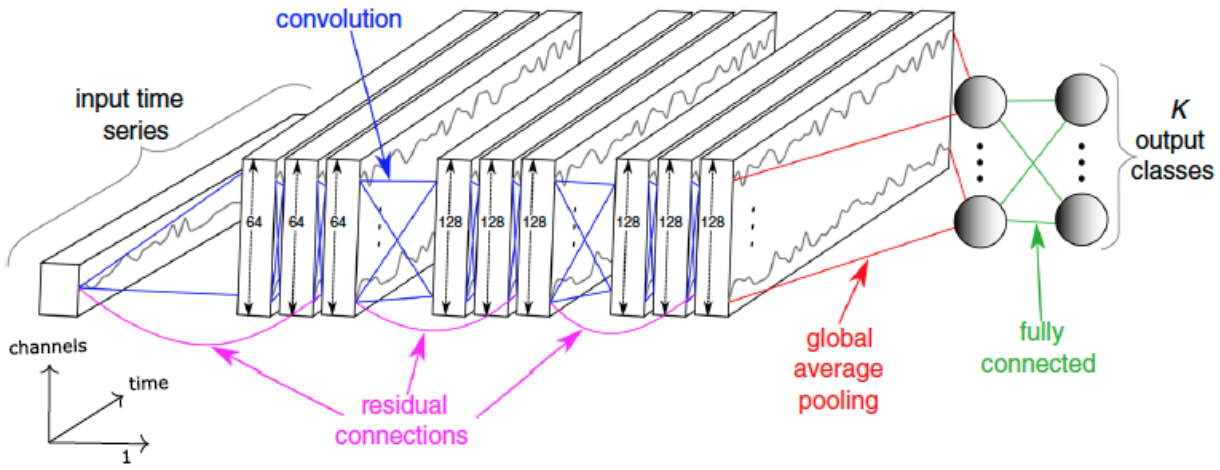


Figure 4. The ResNet architecture. This figure is taken from (*12*) for illustration purposes.

Methods

Approach: Arrhythmia classification from ECG involved analysis of a very long time series, with 5,000 timesteps (500 samples per second taken for 10 sec). The sequential nature of cardiac rhythm made It necessary to model this classification problem as a time series problem.

Dataset

The ECG data was identified from the Chapman-Shaoxing ECG database (*16*, *17*). The 12-lead ECG of 10,646 individuals was recorded at a 500Hz sampling frequency for 10 seconds long. Each patient's recording contains 10 seconds recording, with 500 samples per second, a series of 5,000 samples and 12 features, 1 pear each lead. The dataset includes diagnosis of each patient, including 11 common heart rhythms and 67 other conditions. However, each of the 67 conditions and some of the rhythms are rare individually, and they were combined into 4-categories of arrhythmias as suggested by the data developers (*16*). The four arrhythmia classes are shown in table 1. The denoised ECG data provided as a CSV file is used. The dataset "ECGDataDenoised.zip" is accessible via the link https://figshare.com/collections/ChapmanECG/4560497/2.

Table 2: List of 11 arrhythmia diagnosis and the four categories they were combined into are listed

Acronym	Condition name
SB	Sinus Bradycardia
SR	Sinus Rhythm
AFIB, AFIB	Atrial Fibrillation
ST	Sinus Tachycardia
AF	Atrial Flutter
SI	Sinus Irregularity
SVT	Supraventricular Tachycardia
AT	Atrial Tachycardia
AVNRT	Atrioventricular Node Atrioventricular Node Tachycardia
AVRT	Atrioventricular Reentrant Tachycardia
SAAWR	Sinus Atrium to Atrial Wandering Rhythm

Merged from	4 classes of arrhythmias	Count
AF, AFIB	AFIB	2,225
SVT, AT, SAAWR, ST, AVNRT, AVRT	GSVT	2,307

SB	SB	3,889
SR, SI	SR	2,225
All	All	10,646

The SR category in this study represents normal ECG signals from individuals with no arrhythmias or other abnormal cardiac rhythms.

Data preprocessing

The ECG csv files are denoised and about 1% of the files have 0 entries, and they were excluded from analysis. The CSV files and labels were provided separately, therefore files loaded, and labels were matched carefully.

Prior to modeling, samples were split into train, validation and test (80%, 10%,10%) samples and were normalized using StandardScaler. Labels were encoded with label encoder. Padding was used to make sure that all sequences are the same length. Visualization of ECG samples are included in the jupyter notebook.

Model compiling and training

In this study, 6 models were trained and tested. The six models are: 1D Convolutional neural network (1D CNN), 1D CNN + LSTM, 1D CNN + GRU, ResNet + bidirectional LSTM, InceptionTime, and InceptionTime tuned. All models are compiled and trained using the same method to enable model comparison only with changing models. Learning rate was set at 0.0001 and Adam optimizer was used. A smaller learning rate allows the model to converge more slowly and precisely, minimizing the risk of missing the global minimum. Gradient clipping with Clipvalue=1.0 prevents the gradients from becoming too large. Early stopping was set patience=10, allowing training to stop if no improvement is seen for 10 epochs, essentially preventing overfitting. ReduceLROnPlateau adjusts learning rate decreasing it by a factor of 0.1 if no improvement in validation loss is seen. Epoch was set at 200 and a batch size of 64 was used. Categorical Crossentropy Loss, which is Ideal for multi-class classification is used.

Model evaluation metrics

Identifying ECGs with abnormal rhythms is one of the most crucial aspects of the classification problem, as a missed diagnosis can be detrimental to the patient. Therefore, macro average recall will be used for model comparison. The class-wise precision-recall

curve will also be employed to assess model performance for each class. The area under the precision-recall curve (AUC-PR) is included to obtain a single value that represents how well the model identifies the positive class while minimizing false positives for each label, and this is averaged across all classes. Correctly classifying normal rhythms is also important because false diagnoses can lead to additional clinical tests or treatments, which can be both stressful and costly for patients. Therefore, recall will be monitored to evaluate how well the model identifies the positive groups among those with normal rhythm (sinus rhythm, SR group).

Results

Model 1. 1D CNN model

1D CNN model architecture described in Yildirim et al. (*6*) was selected for this study. The model architecture was chosen because the authors used the same ECG data studied here and demonstrated high accuracy. However, in our approach, the model does not initially include the LSTM layer. The LSTM layer is included in the following model to compare the effects of its inclusion. The model consists of six 1D convolutional layers, along with MaxPooling layers to reduce dimensionality and batch normalization layers to stabilize and accelerate training. The model begins with a series of 1D CNN layers to capture various features and patterns within the input data. The first layer has 64 filters, a large kernel size of 21, and a stride of 11, which helps reduce the input size early on. Subsequent convolutional layers progressively refine the features, increasing the number of filters up to 512 in the network's deeper layers. A LeakyReLU activation function is applied after the first convolution to introduce non-linearity and help mitigate the vanishing gradient problem. The model summary is shown below.

Table 3. 1D CNN model summary

Layer (type)	Output Shape	Param #
conv1d_12 (Conv1D)	(None, 453, 64)	16,192
max_pooling1d_8 (MaxPooling1D)	(None, 226, 64)	0
batch_normalization_4 (BatchNormalization)	(None, 226, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 226, 64)	0
dropout_4 (Dropout)	(None, 226, 64)	0
conv1d_13 (Conv1D)	(None, 220, 64)	28,736
max_pooling1d_9 (MaxPooling1D)	(None, 110, 64)	0
batch_normalization_5 (BatchNormalization)	(None, 110, 64)	256
conv1d_14 (Conv1D)	(None, 106, 128)	41,088
max_pooling1d_10 (MaxPooling1D)	(None, 53, 128)	0
conv1d_15 (Conv1D)	(None, 41, 256)	426,240
conv1d_16 (Conv1D)	(None, 35, 512)	918,016
dropout_5 (Dropout)	(None, 35, 512)	0
conv1d_17 (Conv1D)	(None, 27, 256)	1,179,904
max_pooling1d_11 (MaxPooling1D)	(None, 13, 256)	0
global_average_pooling1d (GlobalAveragePooling1D)	(None, 256)	0
dense_4 (Dense)	(None, 64)	16,448
dense_5 (Dense)	(None, 4)	260

Total params: 7,881,678 (30.07 MB)
Trainable params: 2,627,140 (10.02 MB)
Non-trainable params: 256 (1.00 KB)
Optimizer params: 5,254,282 (20.04 MB)

	precision	recall	f1-score	support
AF	0.96	0.80	0.87	222
GSVT	0.89	0.96	0.92	226
SB	0.96	0.98	0.97	389
SR	0.91	0.95	0.93	222
accuracy			0.93	1059
macro avg	0.93	0.92	0.92	1059
weighted avg	0.93	0.93	0.93	1059

Figure 5. 1D CNN model performance

As shown in figure 5, the model achieved good performance with macro average recall of 0.92 and macro AUC-PR 0.97. This model showed lower performance on classifying AF, as some samples were being classified as GSVT. The model shows slight overfitting as the validation and test performance was lower than train. The following two models will include LSTM and GRU layers to test whether adding these layers on top of the deep CNN models improves performance.

Model 2. CNN + LSTM model

This model is the same as one described in Yildirim et al., The only difference between model 1 and model 2 is the addition of the LSTM layer. The model summary is shown below.

Table 4. Model 2 summary

Layer (type)	Output Shape	Param #
conv1d_18 (Conv1D)	(None, 453, 64)	16,192
max_pooling1d_12 (MaxPooling1D)	(None, 226, 64)	0
batch_normalization_6 (BatchNormalization)	(None, 226, 64)	256
leaky_re_lu_3 (LeakyReLU)	(None, 226, 64)	0
dropout_6 (Dropout)	(None, 226, 64)	0
conv1d_19 (Conv1D)	(None, 220, 64)	28,736
max_pooling1d_13 (MaxPooling1D)	(None, 110, 64)	0
batch_normalization_7 (BatchNormalization)	(None, 110, 64)	256
conv1d_20 (Conv1D)	(None, 106, 128)	41,088
max_pooling1d_14 (MaxPooling1D)	(None, 53, 128)	0
conv1d_21 (Conv1D)	(None, 41, 256)	426,240
conv1d_22 (Conv1D)	(None, 35, 512)	918,016
dropout_7 (Dropout)	(None, 35, 512)	0
conv1d_23 (Conv1D)	(None, 27, 256)	1,179,904
max_pooling1d_15 (MaxPooling1D)	(None, 13, 256)	0
lstm (LSTM)	(None, 128)	197,120
dense_6 (Dense)	(None, 64)	8,256
dense_7 (Dense)	(None, 4)	260

Total params: 8,448,462 (32.23 MB)
Trainable params: 2,816,068 (10.74 MB)
Non-trainable params: 256 (1.00 KB)
Optimizer params: 5,632,138 (21.48 MB)

	precision	recall	f1-score	support
AF	0.93	0.83	0.88	222
GSVT	0.90	0.94	0.92	226
SB	0.97	0.99	0.98	389
SR	0.94	0.95	0.95	222
accuracy			0.94	1059
macro avg	0.94	0.93	0.93	1059
weighted avg	0.94	0.94	0.94	1059

Figure 6. Model 2- CNN + LSTM performance

As shown in Figure 6, model 2 achieved a good performance with macro average recall of 0.93 and macro-AUC-PR 0.98. Including the LSTM layer improved the overall model performance, and on classifying AF. Using bidirectional LSTM instead of LSTM layer showed slight reduction in performance.

Model 3. 1D CNN + GRU

This model structure is the same as the two models as above, but instead of LSTM, it uses GRU layer. The model summary is not included here since it is similar to the ones described above. Both LSTM and GRU are known for their ability to capture longer-range dependencies, but they might not be as effective as LSTMs when dealing with very complex sequence relationships. The LSTM model has more trainable parameters (model 2 total trainable parameters = about 8.5 million) with three gates as opposed to two in GRUs (model 3 total trainable parameters = 1.4 million).

	precision	recall	f1-score	support
AF	0.90	0.80	0.85	222
GSVT	0.89	0.92	0.90	226
SB	0.96	0.98	0.97	389
SR	0.92	0.94	0.93	222
accuracy			0.92	1059
macro avg	0.92	0.91	0.91	1059
weighted avg	0.92	0.92	0.92	1059

Figure 7. Model 3 – CNN + GRU performance metrics

Compared to the CNN + LSTM model, using GRU did not improve performance. The macro average recall decreased from 0.92 to 0.91. Model 3 did not show improvement in performance when compared to the CNN only model. This observation suggests that the LSTM layer is capturing additional features not captured by the CNN layers, while GRU doesn't capture additional features that were not captured by the CNN layers.

Model 4. ResNet + bidirectional LSTM

The simplified ResNet model includes 1D convolutional layers, Residual blocks and Bidirectional LSTM layer. The residual block is the central component of the architecture. This model includes only two residual blocks, instead of 3 described in Wang et al., 2017 (Time series classification from scratch with deep neural networks: a strong baseline)

Model 4 starts with a convolutional layer with ReLU activation and batch normalization, followed by another convolutional layer. The input (shortcut) is added back to the output of these convolutions to form a residual connection, allowing the model to learn identity mappings, which helps in training deeper networks effectively. A ReLU activation is applied

after adding the shortcut. Two of these residual blocks are used in sequential layers. After each residual block, Maxpooling is used to reduce dimensionality. Bidirectional LSTM layer is added to capture additional sequence related features. Return sequence is set to 'False' to ensure that the layer output is a single vector instead of sequence, since this is the last layer before the output layer. Dropout layer was included to reduce overfitting. The ResNet model architecture is shown in figure 8.

Table 5. ResNet + bidirectional LSTM

Layer (type)	Output Shape	Param #	Connected to
input_layer_7 (InputLayer)	(None, 4999, 12)	0	–
conv1d_46 (Conv1D)	(None, 4999, 128)	4,736	input_layer_7[0][0]
batch_normalization_22 (BatchNormalization)	(None, 4999, 128)	512	conv1d_46[0][0]
conv1d_47 (Conv1D)	(None, 4999, 128)	49,280	batch_normalization_2…
batch_normalization_23 (BatchNormalization)	(None, 4999, 128)	512	conv1d_47[0][0]
conv1d_48 (Conv1D)	(None, 4999, 128)	49,280	batch_normalization_2…
batch_normalization_24 (BatchNormalization)	(None, 4999, 128)	512	conv1d_48[0][0]
add_4 (Add)	(None, 4999, 128)	0	batch_normalization_2… batch_normalization_2…
activation_4 (Activation)	(None, 4999, 128)	0	add_4[0][0]
max_pooling1d_28 (MaxPooling1D)	(None, 2499, 128)	0	activation_4[0][0]
conv1d_49 (Conv1D)	(None, 2499, 128)	49,280	max_pooling1d_28[0][0]
batch_normalization_25 (BatchNormalization)	(None, 2499, 128)	512	conv1d_49[0][0]
conv1d_50 (Conv1D)	(None, 2499, 128)	49,280	batch_normalization_2…
batch_normalization_26 (BatchNormalization)	(None, 2499, 128)	512	conv1d_50[0][0]

add_5 (Add)	(None, 2499, 128)	0	batch_normalization_2… max_pooling1d_28[0][0]
activation_5 (Activation)	(None, 2499, 128)	0	add_5[0][0]
max_pooling1d_29 (MaxPooling1D)	(None, 1249, 128)	0	activation_5[0][0]
bidirectional_2 (Bidirectional)	(None, 256)	263,168	max_pooling1d_29[0][0]
dropout_14 (Dropout)	(None, 256)	0	bidirectional_2[0][0]
dense_12 (Dense)	(None, 4)	1,028	dropout_14[0][0]

Total params: 1,403,278 (5.35 MB)
Trainable params: 467,332 (1.78 MB)
Non-trainable params: 1,280 (5.00 KB)
Optimizer params: 934,666 (3.57 MB)

	precision	recall	f1-score	support
AF	0.85	0.82	0.83	222
GSVT	0.81	0.81	0.81	226
SB	0.91	0.96	0.93	389
SR	0.83	0.79	0.81	222
accuracy			0.86	1059
macro avg	0.85	0.84	0.85	1059
eighted avg	0.86	0.86	0.86	1059

Figure 8. Model 4 - ResNet + bidirectional LSTM model

Model 4 did not perform better than the deep 1D CNN model. The macro average recall of 0.84 is much lower than the CNN + LSTM or CNN + GRU models. ResNet with GlobalAveragePooling1D layer, as described previously, instead of LSTM was also tested, but model performance was similar to one with bidirectional LSTM. Adding additional residual blocks might improve performance, however this was not tested. The train and

validation loss plot shows instability in learning rate during the first 10 ephocs, suggesting potential model instability.

Model 5. InceptionTime model

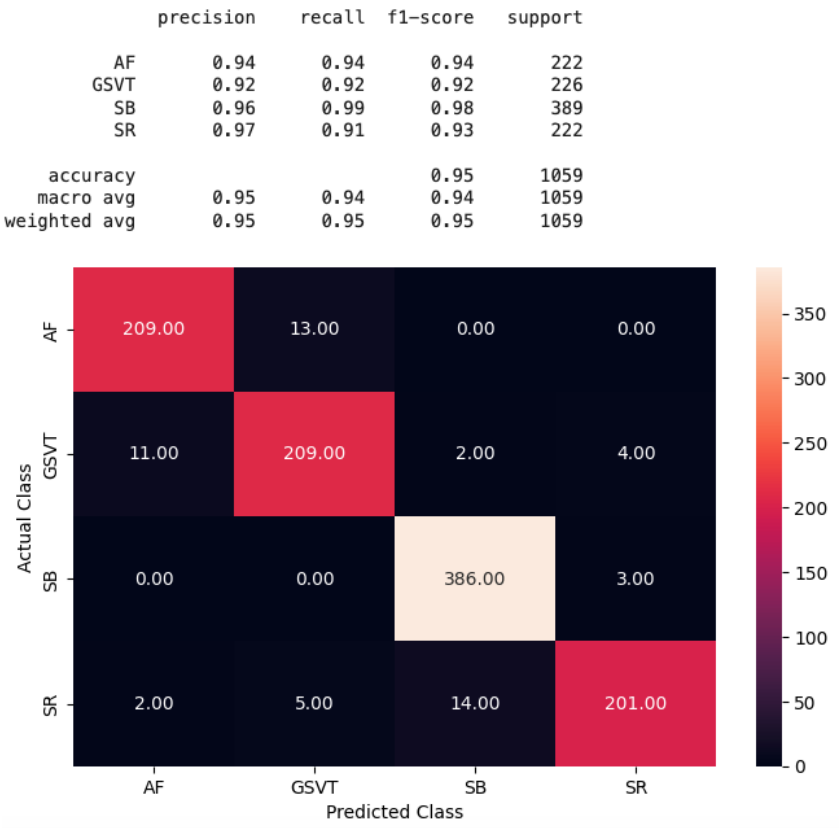
The original model architecture described in Ismail Fawaz et al., 2020 is used. The network includes 6 Inception modules and 2 residual connections. Inception modules encompass three parallel branches with convolutions of different kernel sizes and an average pooling branch. These branches are concatenated.

Table 6. Model 5 - InceptionTime model summary

Layer (type)	Output Shape	Param #	Connected to
input_layer_9 (InputLayer)	(None, 4999, 12)	0	–
conv1d_73 (Conv1D)	(None, 4999, 128)	1,536	input_layer_9[0][0]
inception_module (InceptionModule)	(None, 4999, 128)	115,968	input_layer_9[0][0]
batch_normalization_41 (BatchNormalization)	(None, 4999, 128)	512	conv1d_73[0][0]
inception_module_1 (InceptionModule)	(None, 4999, 128)	123,392	inception_module[0][0]
add_6 (Add)	(None, 4999, 128)	0	batch_normalization_4… inception_module_1[0]…
inception_module_2 (InceptionModule)	(None, 4999, 128)	123,392	add_6[0][0]
conv1d_89 (Conv1D)	(None, 4999, 128)	1,536	input_layer_9[0][0]
inception_module_3 (InceptionModule)	(None, 4999, 128)	123,392	inception_module_2[0]…
batch_normalization_45 (BatchNormalization)	(None, 4999, 128)	512	conv1d_89[0][0]
inception_module_4 (InceptionModule)	(None, 4999, 128)	123,392	inception_module_3[0]…
add_7 (Add)	(None, 4999, 128)	0	batch_normalization_4… inception_module_4[0]…
inception_module_5 (InceptionModule)	(None, 4999, 128)	123,392	add_7[0][0]
global_average_pooling1d… (GlobalAveragePooling1D)	(None, 128)	0	inception_module_5[0]…
dense_14 (Dense)	(None, 4)	516	global_average_poolin…

Total params: 2,208,526 (8.42 MB)
Trainable params: 735,492 (2.81 MB)

The inception module uses 32 filters. Relu activation is applied after each convolution. The first 1D convolutional (conv1) layer uses a kernel size of 1, acting as a bottleneck to reduce dimensions, followed by max pooling layer with a pool size of 3. The three convolutions (conv2, conv3, conv4) are three 1D convolutional layers with increasing kernel sizes (16, 32, and 64, respectively), all using relu activation. The fifth convolution (conv5) is another 1D convolutional layer with a 1x1 convolution to process the output from the max pooling layer. A concatenate layer to merge the outputs from different convolution paths. A batch normalization layer is included to stabilize the learning and improve convergence. The shortcut layer implements a shortcut (or skip) connection for a residual network. It adds the input tensor and the output of the Inception module.



	precision	recall	f1-score	support
AF	0.94	0.94	0.94	222
GSVT	0.92	0.92	0.92	226
SB	0.96	0.99	0.98	389
SR	0.97	0.91	0.93	222
accuracy			0.95	1059
macro avg	0.95	0.94	0.94	1059
weighted avg	0.95	0.95	0.95	1059

Figure 9. Model 5 - InceptionTime model performance

Model 5 performed the best of all the models above, with macro average recall of 0.94 and macro AUC-PR of 0.97. This model showed the least confusion between AF and GSVT classes. Although the model has over 2 million trainable parameters, it ran faster than the CNN + LSTM or GRU models.

Model 6. Tuned InceptionTime model

Hyperparameter tuning was performed for the InceptionTime model. As suggested by Ismael et al., 2020, most important hyperparameters are the number of filters (e.g., 16, 32, 64) and kernel sizes in the three convolution layers within the inception module. Hyperparameter tuning revealed that the default number of filters, which is 32, and kernel sizes 16, 32, and 96 in the three convolution layers are the best parameters. When using these parameters, the model performance improved slightly on some of the class labels but reduced on others.

	precision	recall	f1-score	support
AF	0.93	0.93	0.93	222
GSVT	0.90	0.93	0.92	226
SB	0.96	0.99	0.97	389
SR	0.97	0.87	0.92	222
accuracy			0.94	1059
macro avg	0.94	0.93	0.93	1059
ighted avg	0.94	0.94	0.94	1059

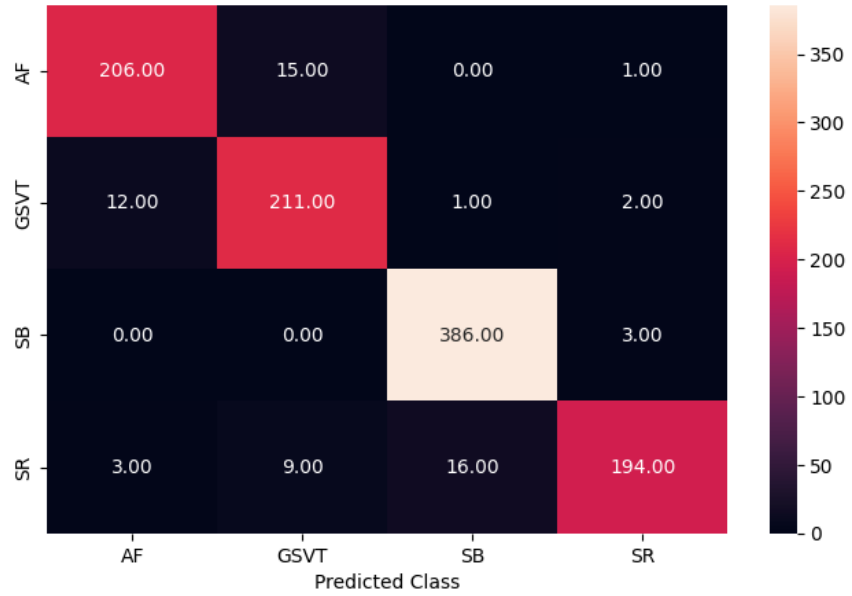


Figure 10. Model 6 – tuned InceptionTime model

Overall, the macro average recall was 0.93, which was less than model 5 score of 0.94.

Model comparison

Table 7. Comparison of macro averaged and class level recall across different models

Labels	CNN	CNN+LSTM	CNN+GRU	ResNet + LSTM	InceptionTIme	InceptionTIme tuned
AF	0.80	0.83	0.80	0.82	0.94	0.93
GSVT	0.96	0.94	0.92	0.81	0.92	0.93
SB	0.98	0.99	0.98	0.96	0.99	0.99
SR	0.95	0.95	0.94	0.79	0.91	0.87
Macro average recall	0.92	0.93	0.91	0.84	**0.94**	0.93

The InceptionTime model performed the best, with macro average recall of 0.94.

Evaluation of misclassified samples revealed that the tuned model confusions occur between GSVT and AF classes.

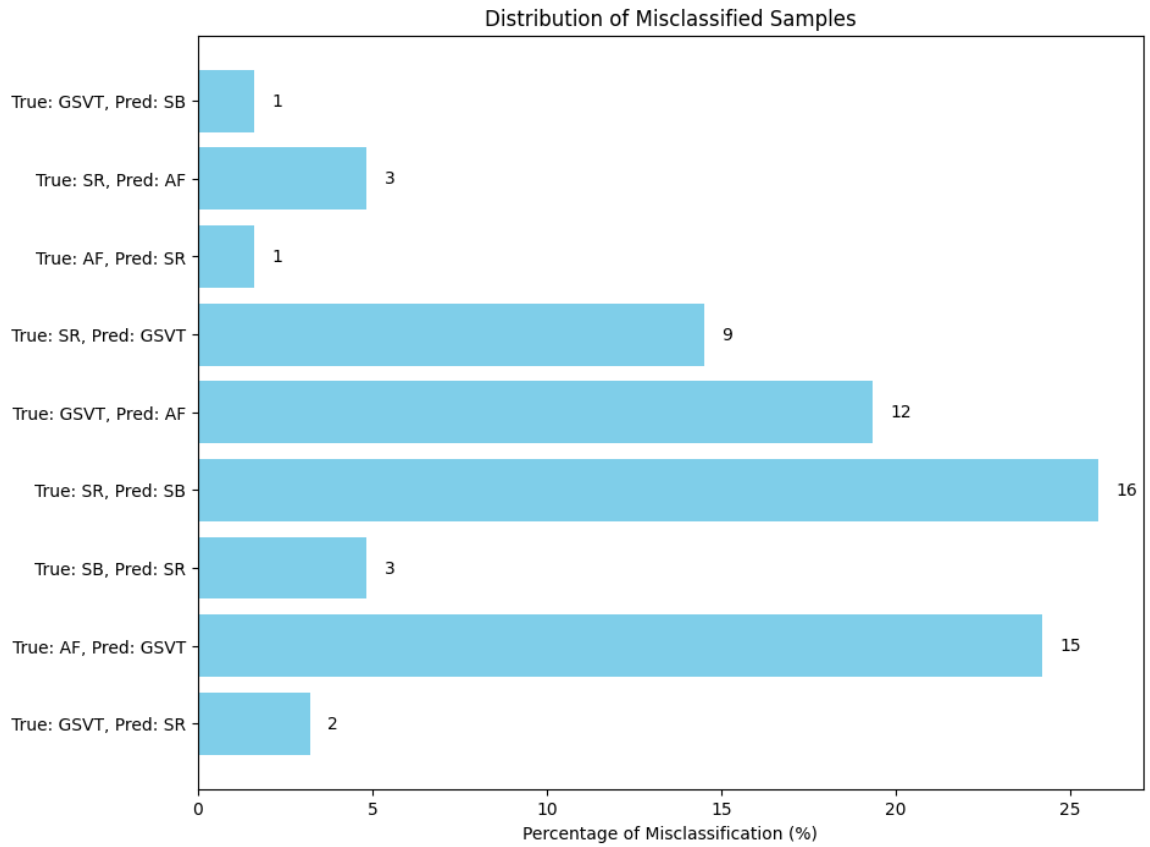


Figure 11. Evaluation of misclassified labels. Percentage of all misclassified samples with each misclassification is shown on the x-axis. The number of misclassified samples are shown on top of each bar.

Twelve GSVT samples are classified as AF and 15 AF are classified as GSVT. The next common confusion is between SR and SB, where 16 SR samples are misclassified as SB. This information gives insights into factors that lead to the model confusion. Notably, all of the models examined consistently showed confusions between AF and GSVT. Therefore a close examination of ECG patterns between these two groups could uncover the reason for this confusion.

Discussion and conclusions

Consistent with previous studies, this report shows that deep learning methods can achieve high performance in ECG classification of arrhythmias. In particular, model 2, which combines 1D CNN and LSTM, as well as model 5, InceptionTime, showed superior performance. In another study (6), the deep 1D CNN + LSTM model showed

an accuracy of 96% on arrhythmia classification using the same ECG data studied here. Consistently, in this report, the model showed an average accuracy of 94%, which is very close to the published results. More importantly, both models 2 and 5 are fairly generalizable, as they are not substantially overfitting. To my knowledge, the InceptionTime model has not been used for ECG classification with Chapman's data, but another study (10) reported an F1 score of 0.92 on arrhythmia classification using ECGs from another database, which is similar to the score of 0.94 in this study.

Interestingly, only the InceptionTime model showed a good balance between correct classification of AF and GSVT groups, while all the deep 1D CNN models performed the worst in classifying these two groups. Perhaps combining the strengths of the CNN models and the InceptionTime model could result in even better performance, and adding CNN layers after InceptionTime modules might lead to further performance improvement. Furthermore, to delve deeper into the AF and GSVT confusion, sequentially removing each of the minority arrhythmia subtypes and observing whether the confusion is reduced could help determine which ECG might be causing the misclassification. This simple analysis could lead to the discovery of which ECG abnormality is responsible for the model confusion. In addition, including age and sex in the prediction could further improve the InceptionTime model performance as they are clinically important.

References

1. Y. Ansari, O. Mourad, K. Qaraqe, E. Serpedin, Deep learning for ECG Arrhythmia detection and classification: an overview of progress for period 2017-2023. *Front. Physiol.* **14**, 1246746 (2023).

2. U. Gupta, N. Paluru, D. Nankani, K. Kulkarni, N. Awasthi, A comprehensive review on efficient artificial intelligence models for classification of abnormal cardiac rhythms using electrocardiograms. *Heliyon* **10**, e26787 (2024).

3. S. Kiranyaz, T. Ince, M. Gabbouj, Real-time patient-specific ECG classification by 1-D convolutional neural networks. *IEEE Trans. Biomed. Eng.* **63**, 664–675 (2016).

4. Y. Li, Y. Pang, J. Wang, X. Li, Patient-specific ECG classification by deeper CNN from generic to dedicated. *Neurocomputing*, doi: 10.1016/j.neucom.2018.06.068 (2018).

5. Ö. Yıldırım, P. Pławiak, R.-S. Tan, U. R. Acharya, Arrhythmia detection using deep convolutional neural network with long duration ECG signals. *Comput. Biol. Med.* **102**, 411–420 (2018).

6. O. Yildirim, M. Talo, E. J. Ciaccio, R. S. Tan, U. R. Acharya, Accurate deep neural network model to detect cardiac arrhythmia on more than 10,000 individual subject ECG records. *Comput. Methods Programs Biomed.* **197**, 105740 (2020).

7. W. Midani, W. Ouarda, M. B. Ayed, DeepArr: An investigative tool for arrhythmia detection using a contextual deep neural network from electrocardiograms (ECG) signals. *Biomed. Signal Process. Control* **85**, 104954 (2023).

8. C. Chen, Z. Hua, R. Zhang, G. Liu, W. Wen, Automated arrhythmia classification based on a combination network of CNN and LSTM. *Biomed. Signal Process. Control* **57**, 101819 (2020).

9. J. H. Tan, Y. Hagiwara, W. Pang, I. Lim, S. L. Oh, M. Adam, R. S. Tan, M. Chen, U. R. Acharya, Application of stacked convolutional and long short-term memory network for accurate identification of CAD ECG signals. *Comput. Biol. Med.* **94**, 19–26 (2018).

10. N. Strodthoff, P. Wagner, T. Schaeffter, W. Samek, Deep learning for ECG analysis: Benchmarks and insights from PTB-XL. *IEEE J. Biomed. Health Inform.* **25**, 1519–1528 (2021).

11. H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, *arXiv [cs.LG]* (2018). http://arxiv.org/abs/1809.04356.

12. H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review. *Data Min. Knowl. Discov.* **33**, 917–963 (2019).

13. O. Yildirim, U. B. Baloglu, R.-S. Tan, E. J. Ciaccio, U. R. Acharya, A new approach for arrhythmia classification using deep coded features and LSTM networks. *Comput. Methods Programs Biomed.* **176**, 121–133 (2019).

14. H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, InceptionTime: Finding AlexNet for time series classification. *Data Min. Knowl. Discov.* **34**, 1936–1962 (2020).

15. Z. Wang, W. Yan, T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline" in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017; http://dx.doi.org/10.1109/ijcnn.2017.7966039).

16. J. Zheng, J. Zhang, S. Danioko, H. Yao, H. Guo, C. Rakovski, A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients. *Sci. Data* **7**, 48 (2020).

17. A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, H. E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* **101**, E215-20 (2000).