AFib Explainable Artificial Intelligence

Shubhkarman Singh
University of Washington
Seattle, Washington, USA
shubhs2@cs.washington.edu

Zeynep Toprakbasti
University of Washington
Seattle, Washington, USA
ztoprakb@cs.washington.edu

Nawaf Osman
University of Washington
Seattle, Washington, USA
nawaf3@uw.edu

Kasper Lindberg
University of Washington
Seattle, Washington, USA
whet@cs.washington.edu

Eyad Alsilimy
University of Washington
Seattle, Washington, USA
alsile@uw.edu

ABSTRACT

Currently, there are many ML models that, given a patient's ECG, can diagnose whether that patient has AFib with high accuracy. The issue with many of these models is that they are black-box models which make it difficult for community healthcare workers to interpret their decisions. Our solution is AFib-XAI, a program that uses explainable AI methods to display the important data points and regions of interest in an ECG that were factored into a model's AFib diagnosis, as well as generate accurate, human-readable explanations for that model's diagnosis. AFib-XAI can be run through a command-line interface. The program offers a selection of 3 AFib diagnostic deep learning models and 4 SHAP-based explainability methods. The user must provide an ECG, make a model selection, and select an explainability method. Afterward, the program will be run. Once the program has finished running, the user will be given a visual result from the SHAP explainability method, as well as a very basic text explanation generated from the results.

KEYWORDS

Atrial Fibrillation, Machine Learning, Explainable Artificial Intelligence, SHapley Additive exPlanations, Electrocardiogram

1 INTRODUCTION

AFib-XAI connects atrial fibrillation (A-Fib) detection neural networks with explainable artificial intelligence (XAI) methods to provide insight on how these black box detection models are making crucial decisions.

1.1 Background

Cardiovascular diseases (CVDs) are the leading cause of death globally.An estimated 17.9 million people died from CVDs in 2019, representing 32% of all global deaths. Of these deaths, 85% were due to heart attack and stroke. [2] Over three quarters of CVD deaths take place in low-income and middle-income countries.

Atrial fibrillation (A-fib) is frequently undetected but carries risks such as stroke, heart failure, and death. A-Fib causes about 1 in 7 strokes. [1] With A-Fib being such a deadly disease, the diagnosis of A-Fib should not be taken lightly. However, the required healthcare costs associated with treating such a diagnosis can be substantial, especially for low-income and middle-income families living in developing countries. The current screening methods are limited by their high cost, low effectiveness, and need for prolonged monitoring. For example, in rural parts of India, there aren't the necessary doctors, let alone healthcare facilities to diagnose and take care of A-Fib patients. In general, if a diagnosis is unclear, it's easier for patients to ignore the diagnosis, for many reasons, including monetary. Thus, a lack of clarity in diagnosis likely contributes to a higher death count relative to other countries because in low-income and middle-income countries, the stakes are much higher.

1.2 Problem Statement

The big issue with diagnosing A-Fib is detection before an episode. It is not so difficult to diagnose a patient during an A-fib episode. However, patients that are likely to experience an episode in the future and healthy patients can have seemingly identical ECG scans. Nonetheless, there exist black box neural networks for diagnosing A-fib before an episode, and these neural networks can pick up on the minuscule differences between healthy and A-Fib patients that the human eye cannot. Though the accuracy of these neural networks is high, due to their black box nature, the reasoning behind their decisions is unknown. The field of XAI seeks to uncover the black box nature of neural networks. Specifically, XAI seeks to provide explanations and reasoning for the decisions that neural networks make. By applying XAI to neural networks that diagnose A-Fib, we hope to help families and community healthcare workers be better equipped to make informed decisions about treating patients by providing accurate explanations and reasoning for their diagnosis.

1.3 Goals

Given current models for diagnosing A-Fib, develop a software which accomplishes the following goals:

- Use XAI to analyze the way these model's detect A-Fib and determine which features the model tends to focus on.
- Use XAI to narrow down a list of good models whose behavior can be understood by the medical community and pushed forward to less experienced workers. These models paired with their XAI data can be used to develop intuition and understanding for those hoping to diagnose A-Fib from these ECGs.

2 DATA PROCESSING

2.1 ECG Matrix Data

To start, from our ECG datasets, we extracted 30,000 sinus rhythm ECGs from patients with A-fib, which we'll refer to as A-fib ECGS, and 30,000 sinus rhythm ECGs from patients without A-fib, as a control, which we'll refer to as control ECGs. Then we create a numpy array of size 30,000 filled in ones, and a numpy array of size 30,000 filled with zeros. We stack the control ECGs on top of the A-fib ECGs and we also stack the zero filled numpy array on top of the one filled numpy array. Next, we generate a random index matrix that contains 60,000 randomly generated numbers ranging from 0 to 59,999. This matrix will be used as an index to select from both X and Y when training the models so that the data is shuffled, but each label will still correspond to the correct ECG type.

2.2 Image Data

Using standard Fourier transforms, we also converted ECGs into 300x300 pixel grey-scale spectrogram images using SciPy's Fourier transform methods as seen in Figure 1 to implement diagnosis models for ECG images rather than ECG matrix data.

3 MODEL SELECTION AND TRAINING

There are three specific models that were used to train on Afib data: Attia et al. CNN Model, ResNet18-Grayscale CNN Model, and an LSTM RNN Model. The first and second models are based on the convolutional neural network architecture, which is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. Attia et al. CNN Model is from the Mayo Clinic, where the authors developed an AI-enabled electrocardiograph (ECG) using a convolutional neural network to detect the presence of atrial fibrillation in normal sinus rhythm using standard 10-second, 12-lead ECGs [3]. The study included patients aged 18 years or older with validated rhythm labels acquired at the Mayo Clinic ECG laboratory between Dec 31, 1993, and July 21, 2017. Patients with at least one ECG showing atrial fibrillation or atrial flutter were classified as positive for atrial fibrillation. This paper was chosen as a jumping point for training models to predict atrial fibrillation from normal and abnormal ECG signals. This paper has been cited by numerous others as a strong baseline model.

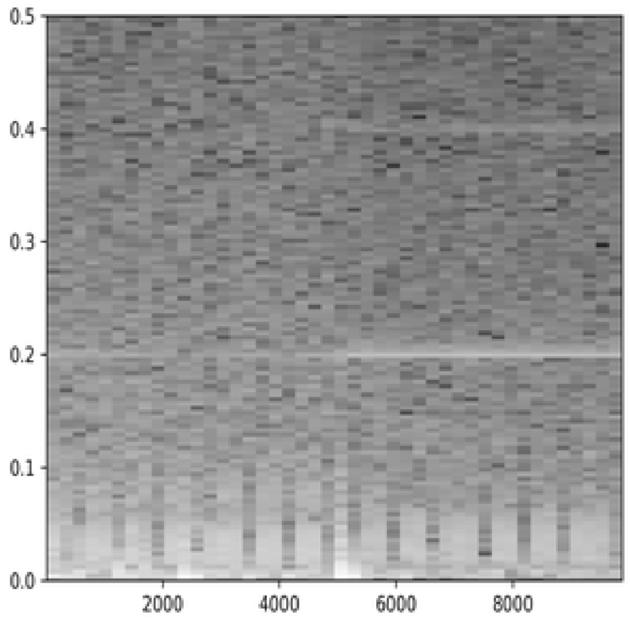


Figure 1: Example ECG Spectrogram

The second CNN model was from a paper that used spectorgrams to dectect structural anomolies. This paper presented a novel approach that combines STFT spectrogram analysis of the ECG signal with handcrafted features to surpass the capabilities of existing commercial products in detecting heart anomalies [6]. By utilizing a Convolutional Neural Network, the proposed algorithm achieves remarkable accuracy in detecting 16 different rhythm anomalies with a 99.79 percent accuracy rate. The main difference from this model and the previous CNN model is the input data. Here we are using the ECG signals transformed into spectorgrams, whereas the previous model used ECG signals. We wanted to provide a more meaningful feature importance representation, that would highlight regions of importance, rather than points of importance. Using spectograms also provides a lesser degree of independence between features (pixels / datapoints) which we want.

The final model we took a look at uses an LSTM model, a recurrent neural network, to address the issue of heavily imbalanced categories in ECG beat data. The proposed method combines a long short-term memory (LSTM) recurrent network model with focal loss (FL) [5]. The LSTM network is capable of capturing timing features in complex ECG signals, while the FL technique is employed to mitigate the category imbalance problem by assigning lower weights to easily identifiable normal ECG examples.

The CNN models utilize weight-sharing to effectively handle spatial data, specifically images that undergo spatial changes. On the other hand, recurrent neural networks (RNNs) are better equipped to capture temporal changes in sequential data, such as the evolving characteristics of sample sequences.

Among the RNN variants, the Long Short-Term Memory (LSTM) network stands out as a popular choice for time series analysis. Its ability to retain historical information and learn long-term dependencies makes it widely applicable in fields like natural language processing, speech recognition, and ECG arrhythmia detection.

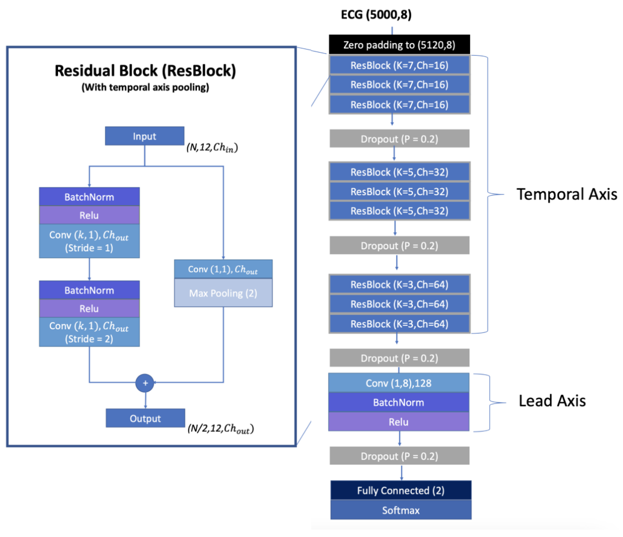


Figure 2: Attia et al. CNN Model Architecture

3.1 Attia et al. CNN Model [3]

The initial CNN model was built using the Keras framework with a TensorFlow backend and implemented in Python. The 12-lead electrocardiogram (ECG) consists of eight physical leads and four augmented leads derived from leads I and II. However, since the augmented leads don't provide additional information, the focus was placed on the eight independent leads (I, II, and V1–6). This selection was made because the models could learn any linear combination of the leads, optimizing performance and reducing the original 12×5000 matrix (12 leads over a 10-second duration sampled at 500 Hz) to an 8×5000 matrix.

Figure 2 provides an overview of the architecture.

The network architecture comprised ten residual blocks, which facilitate direct signal flow to the next layer while preserving the processing within the current layer. This enables the network to learn effectively even when utilizing a large number of layers. Each residual block consisted of two components:

Batch Normalization: This layer normalized the distribution of the data, ensuring stable training by accounting for data normalization.

Non-linear Activation and Convolution: Each block included a non-linear ReLU activation function, which outputs zero for negative inputs and maintains identity for positive inputs. This non-linearity allows the network to create a complex representation of the ECGs, enabling automatic feature extraction.

Additionally, a convolution layer was incorporated to capture local patterns and features. To facilitate gradient propagation, shortcut links were employed. These links used a 1×1 convolutional layer to connect the input of a residual block to its output. Additionally, a max pooling layer was added. These layers were followed by a fusion step, where the outputs of the nine residual blocks, each processing a single lead, were combined using a 1×8 convolutional layer. This merging process integrated the learned representations from all eight independent leads into a unified representation.

After the last convolutional layer, a dropout layer was introduced to prevent overfitting by randomly disabling connections during training. The data then passed through the final output layer, which employed the softmax activation function to generate a probability of atrial fibrillation.

3.2 ResNet18-Grayscale CNN Model [6]

The proposed Convolutional Neural Network (CNN) comprises several key components: a feature-extraction layer, a normalization layer, a CNN block, and a feature-merging layer. The architecture of the proposed CNN is illustrated in Figure 3. In this architecture, the original input is the raw ECG signal, which is used to generate both the STFT spectrogram and handcrafted features. The handcrafted features are then normalized within the range of [0, 1] using the normalization layer to ensure equal contribution from each feature during classification. The STFT spectrogram is passed through the CNN block and a sigmoid layer. The CNN block can be any popular CNN structure, such as VGG-19 CNN, plain 34-layers CNN, or 34-layers Residual Net, and its main purpose is to extract features from the spectrogram using convolutional layers. The sigmoid layer acts as a normalization layer, converting the CNN features to the range of [0, 1], aligning them with the normalized handcrafted features for equal weighting in the final layer. The normalized handcrafted features and the extracted features from the spectrogram are then combined using the feature-merging layer. Finally, the merged feature vector is fed into a fully connected layer to produce the final classification.

For our specific purposes, we modified the CNN model training script developed by Attia et al. to incorporate the CNN block from the proposed CNN architecture. Additionally, we opted to use the 18-layers Residual Net and grayscale spectrograms, as it struck a good balance between model complexity and performance.

Figure 3 provides an overview of the ResNet-18 architecture.

ResNet-18 consists of a convolutional layer and eight residual building blocks, which serve as the fundamental structure of the ResNet-18 network. The structure of the residual building block incorporates a shortcut connection to bypass the convolutional layer. The input vector and the output vector from the convolutional layer are directly added together and passed through the rectified linear unit (ReLU) activation function. This approach effectively mitigates the problem of vanishing or exploding gradients that can occur with deep neural networks.

ResNet-18 encompasses 17 convolutional layers, a max-pooling layer with a specific filter size, and a fully connected layer. A standard ResNet-18 model consists of approximately 33.16 million parameters and employs ReLU activation functions and batch normalization (BN) throughout the convolutional layers in the "basic block."

3.3 LSTM RNN Model [5]

The LSTM network consists of distinct components: an input gate, a forget gate, an output gate, and a cell unit responsible for updating and retaining historical information. Figure 4 illustrates an LSTM block.

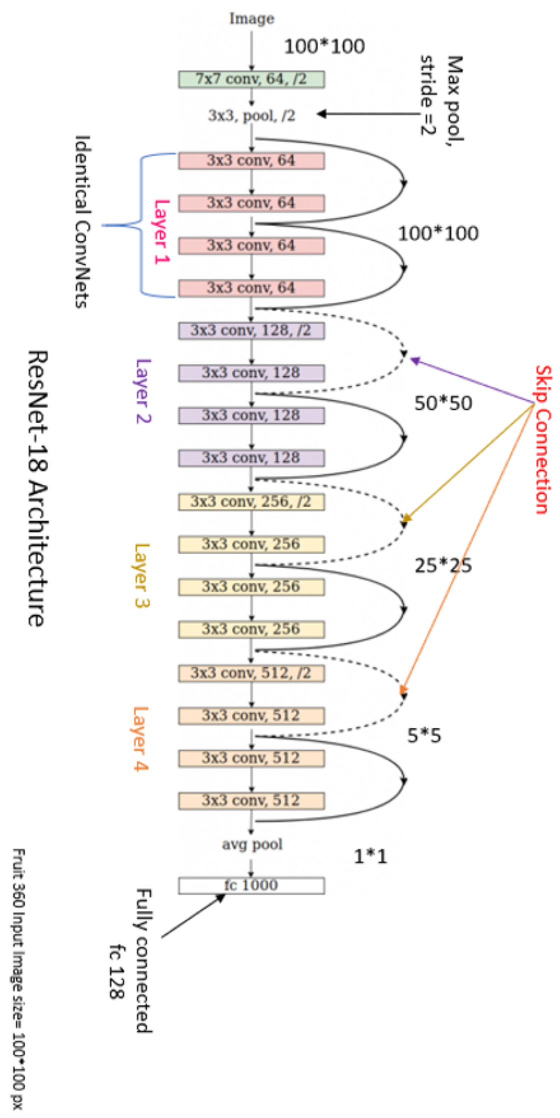


Figure 3: ResNet-18 CNN Model Architecture

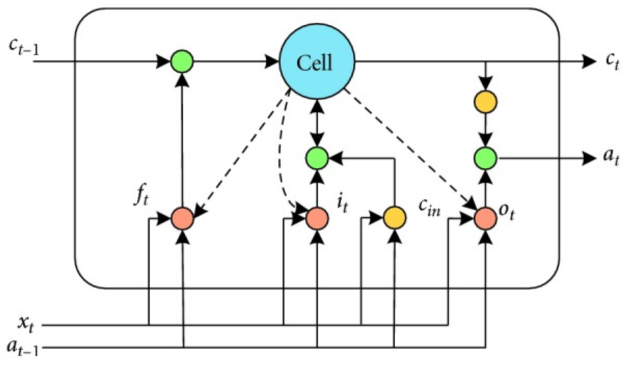


Figure 4: Long short-term memory block

The forget gate within the LSTM memory block is controlled by a single neuron, determining which information should be retained or discarded for preserving historical context. The input gate is generated by a neuron and the effects of the previous memory unit, deciding whether to update the historical information within the LSTM block. A tanh neuron calculates the candidate update content . The current time memory cell state value (ct) is computed using the candidate cell, the previous time state, the input gate, and the forget gate. The output gate generates the output of the

LSTM block at the current time, determining the amount of information from the current cell state to be output.

After computing the hidden vector for each position, we consider the last hidden vector as the representation of the ECG signal. This vector is then fed into a linear layer with an output size equal to the number of classification categories, followed by a softmax output layer to classify the ECG beat as either "N" or "AFIB."

Figure 5 provides an overview of the LSTM model architecture.

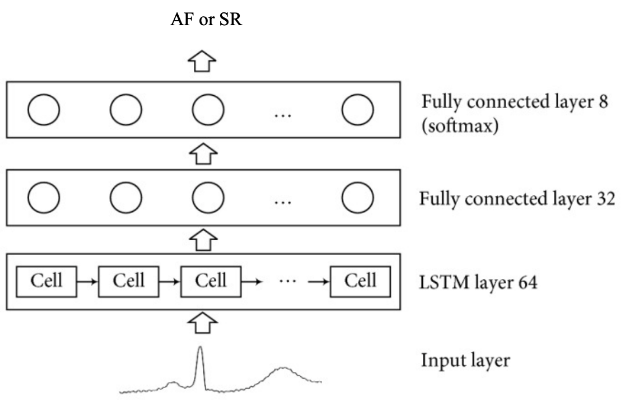


Figure 5: LSTM Model Architecture

In this paper, we employ a four-layer LSTM architecture, consisting of an input layer, an LSTM layer, and two fully connected layers. To address imbalanced datasets, we utilize the focal loss, a more effective approach than traditional cross-entropy (CE) loss. Focal loss transforms the CE loss function. Additionally, the model utilizes the Nadam gradient descent optimization algorithm, which combines the benefits of the Adam and Nesterov Accelerated Gradient (NAG) algorithms to compute adaptive learning rates for different parameters. Overall, Nadam outperforms other gradient descent optimization methods in practical applications.

3.4 Training Procedure

All models were trained on the nigiri cluster with a single NVIDIA GeForce RTX 2080 Ti graphics processing unit that was used to train the models using Keras. We used a Colaboratory Notebook to run a condensed version of the training script to test the models on a small dataset (300 images and 3000 ECG plots), after which, we wrote a polished script through Python. The training script was broken down into these parts: data pre-process, data fetch, train model, save model, load model, and calculate test accuracy. For more information on how the data is processed please look back at Section 2. When training the model, we build the model using the model architecture described in the previous section in a separate Python file. We also employ the use of the ReduceLROnPlateau function to reduce the learning rate when the training validation loss plateaus. The number of epochs, batch size, learning rate, channels, etc. are specified in the training script. Additional save and load functions allow us to use the trained models to perform downstream tasks.

4 EXPLAINABILITY METHODS

Several explainable AI (XAI) methods were tested on the different models. These included mainly SHAP based methods and integrated gradients. The ECG data is a time series data, while the most common use of these methods is in tabular or image based data. However, XAI methods that were better suited towards time series data were often too new in the literature, meaning there weren't code-bases or well documented implementations to follow. Though there was an attempt to implement some time series heuristics presented in these newer papers, the results seemed noisier and less insightful than the standard SHAP based methods.

4.1 SHAP Methods

SHAP values, short for SHapely Additive exPlanations, aims to compute a notion of feature importance by testing how sensitive the model to variability in the feature. Formally, to compute the importance of the ith feature, let N be the number of features and $v : 2^{[N]} \to \mathbb{R}_{\geq 0}$ be the model output based on a masking so for a subset $S \subseteq N$, so $v(S)$ is equal to the model output when the features in S are revealed while the features in $N \setminus S$ are masked by some default value. Then to calculate the ith feature's importance we compute the sum

$$\frac{1}{n} \sum_{S \subseteq N \setminus \{i\}} \binom{|N| - 1}{|S|}^{-1} (v(S \cup \{i\}) - v(S))$$

So the importance is a weighted average over how the model changes when including the ith feature, based on all possible maskings of other features. Note that this is an exponential sum and thus, is rarely practical for our size of data.

To make this computation tractable, a form of approximation is needed. A standard method to do this is approximation is known as KernelSHAP, which instead of computing all possible feature masking subsets, chooses only a polynomial number of these subsets [4]. Then we solve for a linear model using weighted least squares optimization to approximate the whole computation via this partial computation.

It's provable that the approximation reaches the actual SHAP value computation as we increase the number of masked subsets we try. However the quality of this approximation at a tractable number of subsets is usually best understood when the model itself is locally linear around a given feature and is feature independent.

For implementation of KernelShap we use the methods implemented by the SHAP Python library. Our background data, which is used to replace the masked subsets, is a random selection of normal sinus rhythm ECGs. While this form of background data isn't ideal, as we could be replacing parts of the explained ECG with other non-corresponding parts of the background ECG, this yielded better results than just substituting in a zero value.

4.2 Other Explainability Methods

Feature independence is a poor assumption to make in time series data as ECGs model continuous data. One attempt to aid the time series nature of the data was to implement the time-consistent Shapley values from [8], which instead of choosing a tractable number of arbitrary subsets to mask, we instead choose subsets of features corresponding to whole intervals of time. This would hopefully make computations a lot more tractable and better utilize the time dependency, however the importance computations tended to take the same time as KernelShap and have noisier results with less apparent insightfulness.

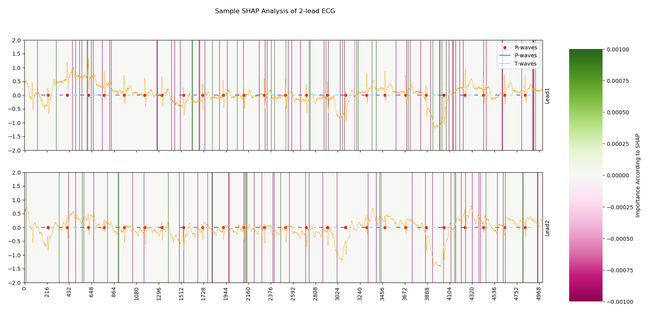


Figure 6: KernelShap values on top and Time-Consistent Shap on bottom

We also experimented with Integrated Gradients as another XAI method [7]. This method also takes a default background ECG and then looks at a linear interpolation between the background ECG and the explained ECG. It then calculates how sensitive each model feature is along this interpolation through gradient methods and gathers a feature importance value from this calculation. This method is again more suited for tabular or image data, and the results of the feature importance were too noisy and widespread to be interpretable.

For the spectrogram data, we also implemented a version of KernelShap onto this image data. We did it similar to KernelShap for the time series data using random ECGs converted into spectrograms as the background mask data. This was computable but highlights on the spectrogram data were not as insightful as time based features that marked the different parts of the ECG.

5 DEVELOPMENT OF THE DELINEATOR

5.1 Background

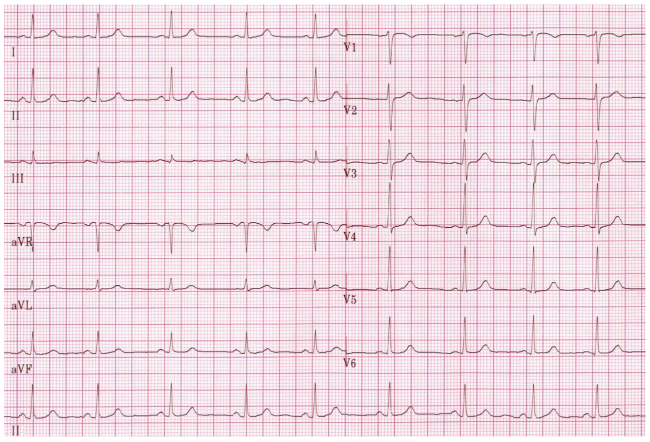


Figure 7: 12-Lead Sinus Rhythm ECG

An electrocardiogram (ECG or EKG) measures the electrical activity of the heart through electrodes placed on the body. The signals and measurements obtained from these electrodes are referred to as leads. One commonly used lead system is the 12-lead system, which consists of leads I, II, III, aVF, aVR, aVL, V1, V2, V3, V4, V5 and V6, as seen in Figure 7. These leads are derived using 10 electrodes. However, some devices such as the KardiaMobile ECG monitor can capture 6 leads (leads I, II, III, aVL, aVR and aVF) using only 3 electrodes. Each lead provides specific information and observable characteristics that may not be available from other leads. Together, they provide a comprehensive 3-D representation of the heart's activity.

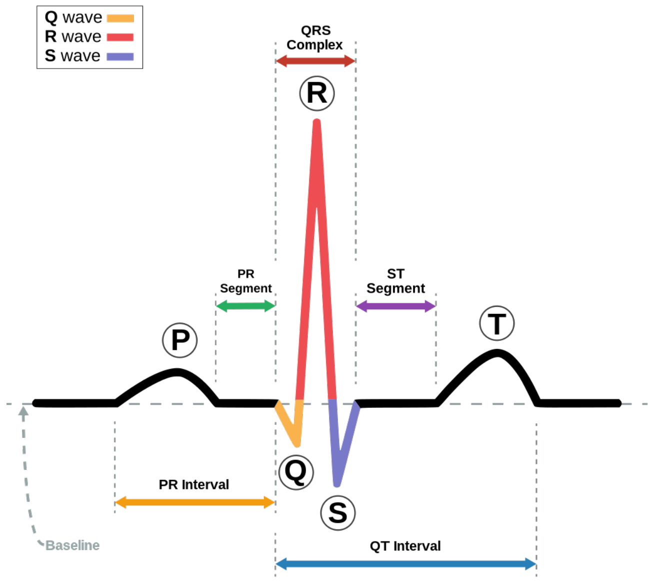


Figure 8: Sinus Rhythm ECG Components

Each ECG lead consists of repeated cardiac cycles, which can be divided into 8 recognized components: the P-wave, PR-segment, Q-wave, R-wave, S-wave, ST-segment, T-wave and TP-segment, as seen in Figure 8. These components represent specific aspects of the heart's activity during a cardiac cycle and can be analyzed individually or collectively to differentiate between a healthy sinus rhythm and an irregular or diseased heart. In the context of this study, the focus is on hearts afflicted with atrial fibrillation (AFib). Not all components may be distinguishable in every lead, which highlights the importance of using a comprehensive lead system.

5.2 Purpose

Delineating a multi-lead ECG into its individual components is crucial for generating accurate and concise text explanations of the output from our explainability methods. The output of our explainability methods will include important data points and regions of interest in the leads of the input ECG that the explainability method recognizes as contributing to the model's diagnosis. These data points and regions of interest will be associated with the ECG components they occupy, highlighting the need for a delineator. The delineator will take a multi-lead ECG as input and return a map of all the components to their positions in their respective leads. For this study, the delineator will identify 6 main components in a multi-lead ECG: P-waves, PR-segments, QRS-complexes (comprising of Q, R and S waves), ST-segments, T-waves and TP-segments.

5.3 Input

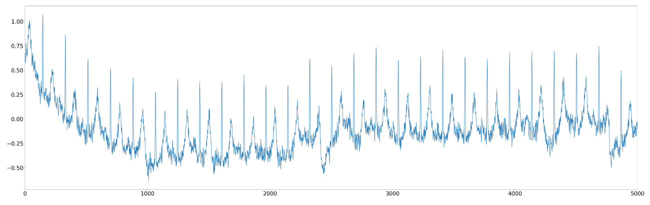


Figure 9: Example Lead II Plot

The delineator takes a 2-D matrix representing the multi-lead ECG as input. Each row of the matrix corresponds to one of the ECG leads, and each column index represents the time position of the signals in milliseconds. To accurately detect all the listed ECG components (P-waves, PR-segments, QRS-complexes, ST-segments, T-waves and TP-segments), the delineator only requires leads I and II.

5.4 ECG Filtering

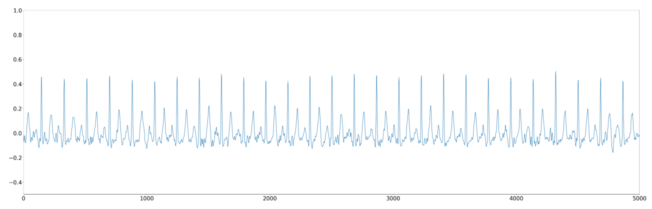


Figure 10: Example Lead II Plot Filtered

Prior to detecting ECG components, leads I and II of the input ECG must be filtered to remove excessive noise and artifacts that could impact the delineators' ability to detect these components or cause false positive detections. Two types of noise are filtered out: baseline wander and power-line interference.

Baseline wander is an effect where the base axis of a signal appears to wander rather than remain relatively straight. Since the baseline signal is low frequency, to filter out baseline wander, a high-pass Butterworth filter with a cutoff frequency of 3 hertz and an order of 2 is applied to both leads, resulting in relatively straight signals.

Next, electromagnetic fields caused by power-lines are a common source of noise that can hinder the detection of ECG components. Spurious wave-forms may be introduced to the signals and low frequency waves like the P or T waves may be superimposed. Since power-line interference is narrow-band noise with a frequency of either 50 or 60 hertz, To filter out power-line interference, Neurokit's power-line filter with a frequency of 50 hertz, the standard in both the US and

India, is applied to both leads, resulting in smoother, cleaner signals.

After these 2 filters are applied, each lead will be straighter, smoother, and cleaner, as seen in Figure 10 in comparison to the input ECG seen in Figure 9.

5.5 ECG Component Detection

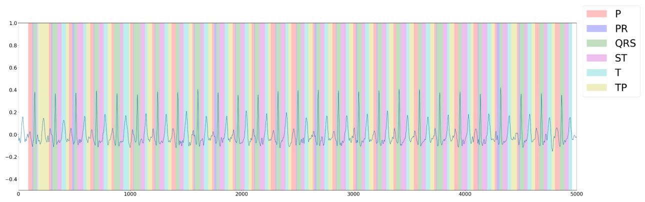


Figure 11: Example Lead II Plot Filtered and Delineated

After leads I and II of the ECG have been filtered, it is time to start the ECG components detection process. The first step is to get the location of the R-wave peaks in lead I. It's important to note that inverted leads can cause inaccurate detections of R-wave peaks, so we use Neurokit2's inverted lead detector to determine whether a lead is inverted, and if it is, we negate the lead, resulting in an upright, normal lead. To get the locations of the R-wave peaks in lead I, we use Neurokit2's peak finder algorithm, passing in the unfiltered lead I signal. Next, we correct the locations of the R-wave peaks to fit lead II using BioSPPy's R-peak correction algorithm since there may be small differences in the locations of the R-wave peaks between the two leads. Lead II has more distinguishable P-waves than lead I, so we will delineate the components using lead II. Now that we have the locations of the R-wave peaks in lead II, we can use Neurokit2's ECG delineate method, passing in the filtered lead II and locations of the R-wave peaks, to get the start and end timestamps of the P-waves, QRS-complexes, and T-waves. We can calculate the positions of the PR-segments as the end timestamps of the P-waves to the start timestamps of the QRS-complexes. Likewise, the positions of the ST-segments will be the end timestamps of the QRS-complexes to the start timestamps of the T-waves. Finally, the positions of the TP-segments will be the end timestamps of the T-waves to the start timestamps of the P-waves.

5.6 Output

The start and end timestamps of each individual component will be stored as a tuple. The delineator will return a map of each ECG component class (P-waves, PR-segments, QRS-complexes, ST-segments, T-waves and TP-segments) mapped to a list of tuples of their respective positions in the ECG.

6 DEVELOPMENT OF THE TRANSLATOR

6.1 Purpose

To reiterate, the output of the implemented SHAP methods includes important data points and regions of interest in the leads of the input ECG that the SHAP method recognizes as contributing to the model's diagnosis. These data points and regions of interest will be associated with the ECG components they occupy, and accurate, concise text explanations will be generated based on those associations. This is the role of the translator; explaining the relevant SHAP values of the ECG leads in an understandable and intuitive form. To do this, the translator will identify clusters of positive SHAP values, which in the case of a positive AFib diagnosis from the model, will be regions of interest in the leads supporting the model's AFib positive diagnosis. The translator will then associate these clusters with their respective ECG components, which would imply abnormalities in those components that are causing the positive diagnosis.

6.2 Input

The translator will take in 3 inputs: a 2-D matrix returned from a SHAP explainability method, containing the SHAP values for each lead in a particular ECG, a map returned from the delineator, containing the positions of all components in the same ECG, and an integer representing cluster sensitivity, which has a default value of 5.

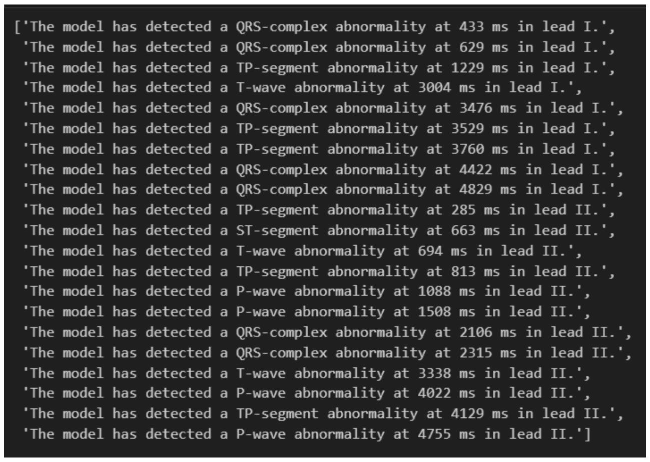
6.3 SHAP Cluster Detection

To detect clusters of positive SHAP values, for each lead in the 2-D matrix, we will check every unique sub-array of size 100, since the QRS-complex typically has a length of about 100 ms and happens to be the ECG component with the greatest length, for a number of positive SHAP values greater than or equal to our given cluster sensitivity variable. We will now define a cluster as a number of positive SHAP values greater than or equal to our given cluster sensitivity variable within a time-frame of 100 ms. If a cluster is not found in a sub-array, we simply move on to the next iteration, checking the next sub-array of size 100. If a cluster is found in a sub-array, we begin the cluster-component association process, after which the next iteration will check another sub-array of size 100, only this time, it will check starting at the first index after the previous sub-array. This is done to reduce the possibility of repeatedly detecting the same cluster.

6.4 Cluster-Component Association

Once a cluster has been found, we need to associate it with the ECG component that it is occupying. To do this, we iterate through the given map returned by the delineator, which contains the timestamps of all the components in the ECG associated with the given SHAP values, and check whether 50% or more of the SHAP values in the cluster are contained within a particular component. If 50% or more of the SHAP values in the cluster are not contained within a particular component, this means that we cannot associate this cluster with one component. If 50% or more of the SHAP values in the cluster are contained within a particular component, we get the index of the first SHAP value of the cluster that is within the particular component and append a sentence containing the particular component and the location of the abnormality to a list of sentences, which we will later return.

6.5 Output



```
['The model has detected a QRS-complex abnormality at 433 ms in lead I.',
 'The model has detected a QRS-complex abnormality at 629 ms in lead I.',
 'The model has detected a TP-segment abnormality at 1229 ms in lead I.',
 'The model has detected a T-wave abnormality at 3004 ms in lead I.',
 'The model has detected a QRS-complex abnormality at 3476 ms in lead I.',
 'The model has detected a TP-segment abnormality at 3529 ms in lead I.',
 'The model has detected a TP-segment abnormality at 3760 ms in lead I.',
 'The model has detected a QRS-complex abnormality at 4422 ms in lead I.',
 'The model has detected a QRS-complex abnormality at 4829 ms in lead I.',
 'The model has detected a TP-segment abnormality at 285 ms in lead II.',
 'The model has detected a ST-segment abnormality at 663 ms in lead II.',
 'The model has detected a T-wave abnormality at 694 ms in lead II.',
 'The model has detected a TP-segment abnormality at 813 ms in lead II.',
 'The model has detected a P-wave abnormality at 1088 ms in lead II.',
 'The model has detected a P-wave abnormality at 1508 ms in lead II.',
 'The model has detected a QRS-complex abnormality at 2106 ms in lead II.',
 'The model has detected a QRS-complex abnormality at 2315 ms in lead II.',
 'The model has detected a T-wave abnormality at 3338 ms in lead II.',
 'The model has detected a P-wave abnormality at 4022 ms in lead II.',
 'The model has detected a TP-segment abnormality at 4129 ms in lead II.',
 'The model has detected a P-wave abnormality at 4755 ms in lead II.']
```

Figure 12: Example Translator Output

The translator will output a list of component abnormalities implied by positive SHAP clusters translated into concise and accurate text explanations. These text explanations will be in the form of sentences following the format seen in Figure 12.

7 FINAL EXPLAINABILITY PIPELINE SOFTWARE ARCHITECTURE

AFib-XAI allows users to select patient's electrocardiogram (ECG) scan, a detection model that will output whether it detects AFib in this ECG, and an XAI model to analyze the selected detection model. AFib-XAI uses these input to produce photo and text results of the XAI model by building the XAI model using the given ECG and detection model. The photo produced is a graph of the given ECG with the XAI findings overlayed on the ECG. The text results translate the graph to specify at what timestamps the XAI model detected a relevant data point in the ECG.

7.1 GUI

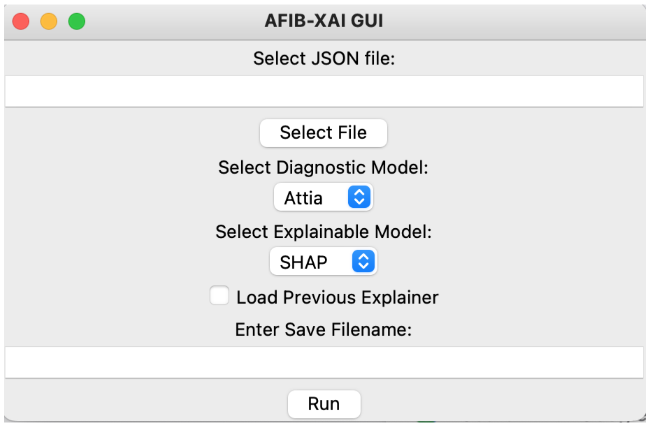


Figure 13: Our program GUI

The GUI pictured in Figure 13 runs AFib-XAI and allows users to specify the filename of the plot result, as well as choose to load a previous XAI model. Building the XAI can take an upwards of 20 minutes. However, if the XAI model is already built, AFib-XAI can produce results in about 15 seconds.

7.2 Pipeline

The terminal command is run with the necessary flags (either directly in the terminal or using the GUI). The program begins by printing an introduction

```
print("Running analysis on the
following parameters...")
print("Model: ", args.model)
print("Explainability Method: ",
     args.xai)
print("Patient ECG File: ", args.ecg)
print("Save Plot To: ", args.save)
```

args is an object containing all of the parameters passed in from the user. So after the user passes in the input parameter, the application prints out what it received as inputs, allowing the user to restart the application if they noticed they made a mistake.

Afterwards, it begins to look through the parameters. If the user would like to use SHAP as its explainability method, the application creates an instance of a SHAP_Explainer and will either load the explainer or build it using buildExplainer(), contained in SHAP_script.py.

After the explainer is either loaded or built, the application pulls the SHAP values from it and prints out the filename they've been saved to. Then it calls plotShap() to generate a plot over the given patients ECG scans with all the SHAP results over it.

8 LIMITATIONS

8.0.1 SHAP considers each data point independently. As described earlier, SHAP works by replacing each data point with a new data point and observing how this modified input changes the result of the detection model. This helps us understand which data points are more significant in the detection model's decision making process. However, we know that with ECGs, the data surrounding any given data point affects the abnormality of the data point of interest. By examining each data point independently, SHAP doesn't consider the regions of the data together, which is relevant since ECG components are regions of a signal.

8.0.2 ECGs are not necessarily synced. Suppose patient A has a heart rate of 60 beats per minute (BPM) and patient B has a heart rate of 90 BPM. If we consider ECG scans from patients A and B, we'll notice that after 1 second in the scan, patients A and B are likely in two completely different ECG components. Patient A's signal may be within a QRS-complex, while patient B's signal may not. Now suppose SHAP is examining patient A, and is using patient B to randomly replace the data point at 1 second into the signal. This could affect the results of the detection model because we have distorted the QRS-complex with a data point taken from a different component. The model may think this is an abnormality because it assumes that the data is all in sync, and using SHAP, this may not necessarily be true.

9 ACKNOWLEDGMENTS

Thank you Richard and Ana for the help and support!

REFERENCES

[1] 2022. Centers for Disease Control and Prevention: Atrial Fibrillation. https://www.cdc.gov/heartdisease/atrial_fibrillation.htm

[2] 2022. World Health Organization: Cardiovascular diseases (CVDs). https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[3] Zachi I Attia, Peter A Noseworthy, Francisco Lopez-Jimenez, Samuel J Asirvatham, Abhishek J Deshmukh, Bernard J Gersh, Rickey E Carter, Xiaoxi Yao, Alejandro A Rabinstein, Brad J Erickson, Suraj Kapa, and Paul A Friedman. 2019. An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction. *The Lancet* 394, 10201 (2019), 861–867. https://doi.org/10.1016/S0140-6736(19)31721-0

[4] Ian Covert and Su-In Lee. 2021. Improving KernelSHAP: Practical Shapley Value Estimation via Linear Regression. arXiv:2012.01536 [cs.LG]

[5] J. Gao, H. Zhang, P. Lu, and Z. Wang. 2019. An Effective LSTM Recurrent Network to Detect Arrhythmia on Imbalanced ECG Dataset. *J Healthc Eng* 2019 (2019), 6320651.

[6] H. Li and P. Boulanger. 2022. Structural Anomalies Detection from Electrocardiogram (ECG) with Spectrogram and Handcrafted Features. *Sensors (Basel)* 22, 7 (Mar 2022).

[7] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. arXiv:1703.01365 [cs.LG]

[8] Mattia Villani, Joshua Lockhart, and Daniele Magazzeni. 2022. Feature Importance for Time Series Data: Improving KernelSHAP. arXiv:2210.02176 [cs.LG]

Received 09 June 2023