Pattern Recognition
Heartbeat Classification

-

-

-

-

Rabbani Nur Kumoro
(21/472599/PA/20310)

Computer Science Study Program
Department of Computer Science and Electronics
Faculty of Mathematics and Natural Science
Universitas Gadjah Mada
Yogyakarta
2023

I. Introduction

Electrocardiogram (ECG) signals are a vital diagnostic tool used in cardiology to monitor and diagnose heart anomalies and diseases. ECG signals are one-dimensional time series signals that represent the voltage changes generated by the electrical activity of the heart over time. The signals are recorded non-invasively using a series of electrodes attached to the patient's chest, arms, and legs, which capture the electrical activity of the heart.

ECG signals are widely used in clinical settings to diagnose a variety of cardiac conditions, including arrhythmias, myocardial infarctions, heart failure, and other abnormalities. The analysis of ECG signals typically involves detecting and characterizing the different components of the signal, such as the P wave, QRS complex, and T wave, which correspond to different phases of the cardiac cycle.

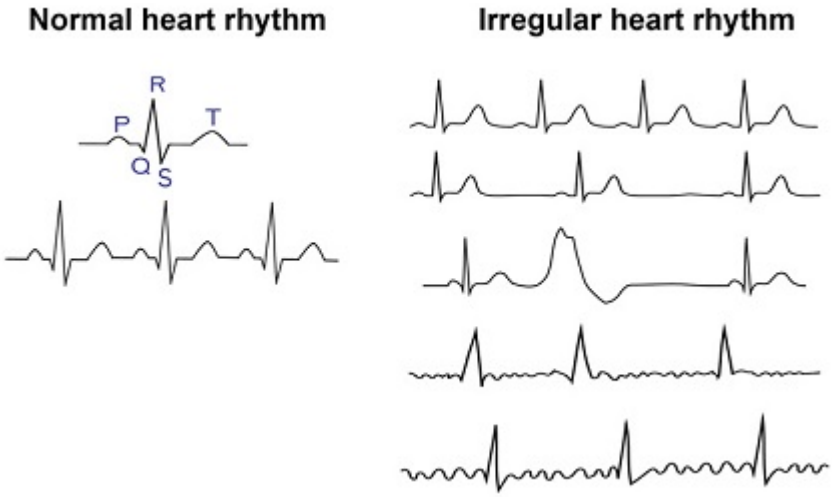


Figure 1. Comparison between Normal and Irregular Heart Beat

Arrhythmia refers to a condition where the heartbeat is irregular or too fast/slow as we can see from **Figure 1**. It can be caused by various factors such as heart attack, stress, smoking, and certain medications. Accurately identifying the type of arrhythmia is crucial for effective treatment and restoring a normal heart rhythm.

In this report, I present a novel approach to ECG heartbeat classification using neural networks. The proposed method is designed to accurately classify five different arrhythmias, including normal beats, supraventricular ectopic beats, ventricular ectopic beats block, fusion beats, and an unknown beat. The neural network is trained using a dataset of labeled ECG signals, with a focus on achieving high accuracy in arrhythmia classification. The results demonstrate the effectiveness of the proposed approach in accurately classifying the different types of arrhythmias, which can aid in the diagnosis and treatment of heart conditions.

II. Methodologies

- Data Acquisition

This dataset was acquired from Kaggle, and it is composed of two collections of heartbeat signals derived from two famous datasets in heartbeat categorization, the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. The number of samples in both collections is large enough for training a deep neural network. It has been widely used for exploring heartbeat classification using deep neural network architectures and investigating the capabilities of transfer learning. The signals in this dataset correspond to electrocardiogram (ECG) shapes of heartbeats for both normal and abnormal cases, including different arrhythmias and myocardial infarctions. Before being included in the dataset, the signals were preprocessed and segmented, with each segment corresponding to a heartbeat.

We will utilize the Arrhythmia Dataset which is a collection of heartbeat signals obtained from the MIT-BIH Arrhythmia Dataset. The dataset contains various types of arrhythmia, which can be caused by several factors such as heart attack, smoking, congenital heart defects, and stress, among others. The identification of the type of arrhythmia is crucial for proper treatment to restore a normal heart rhythm. We will use this dataset to train and test our proposed model for the classification of heartbeats.

MIT-BIH Arrhythmia Dataset Details

Number of Samples: 109446
Number of Categories: 5
Sampling Frequency: 125Hz

‘N’: 0	Non-Ectopic Beats (Normal Beats)
‘S’: 1	Supraventricular Ectopic Beats
‘V’: 2	Ventricular Ectopic Beats
‘F’: 3	Fusion Beats
‘Q’: 4	Unknown Beats

Table 1. Labeled Classes of the Dataset

- Preprocessing

The creator of the Dataset on Kaggle, Shayan Fazeli of UCLA, has pre-processed the dataset using a straightforward yet efficient method for extracting beats from ECG signals. The proposed method involves several steps, as illustrated in **Figure 2**

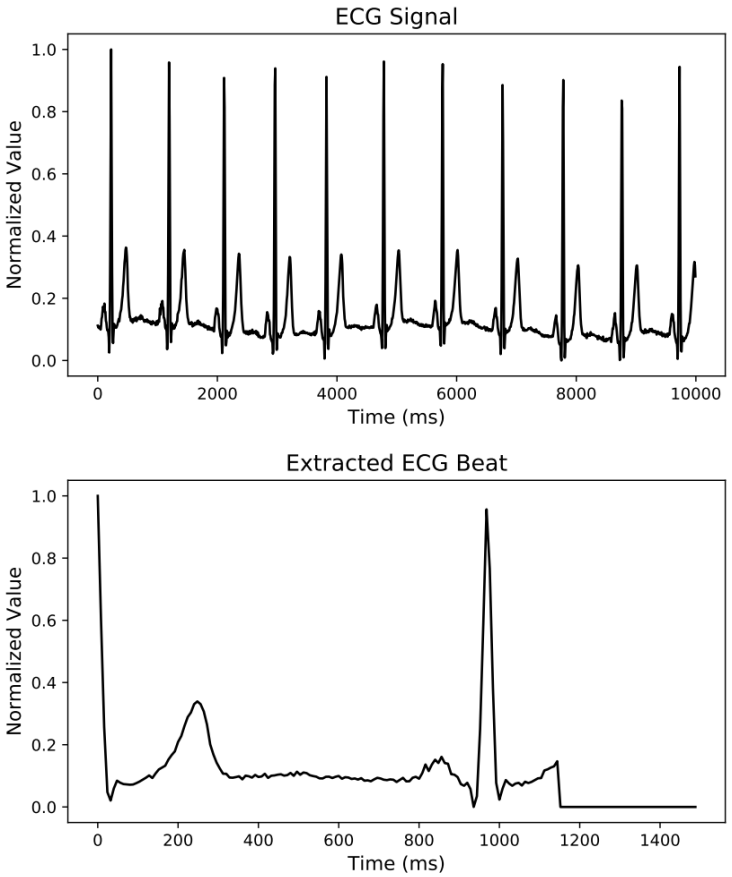


Figure 2. 10s ECG Window and Extracted ECG Beat.

First, the continuous ECG signal is divided into 10-second windows, and one window is selected for analysis. The amplitude values of the window are then normalized to a range of zero to one. The set of local maxima is identified based on the zero-crossings of the first derivative of the signal. A threshold of 0.9 is applied to the normalized values of the local maxima to identify the set of ECG R-peak candidates. The median of R-R time intervals is calculated as the nominal heartbeat period of the window (T).

For each R-peak, a signal part with a length equal to 1.2T is selected. To ensure that all extracted beats have identical lengths, each selected part is padded with zeros to make its length equal to a predefined fixed length.

It is important to note that the proposed beat extraction method is simple yet effective in extracting R-R intervals from signals with different morphologies. No filtering or processing that assumes any signal morphology or spectrum has been employed. Moreover, all extracted beats have identical lengths, which is essential for their use as inputs in subsequent processing stages.

- Exploratory Data Analysis

From **Figure 3** we must take into account that our classes, on which we will apply the classification algorithms, are unbalanced with a majority of normal heartbeats.

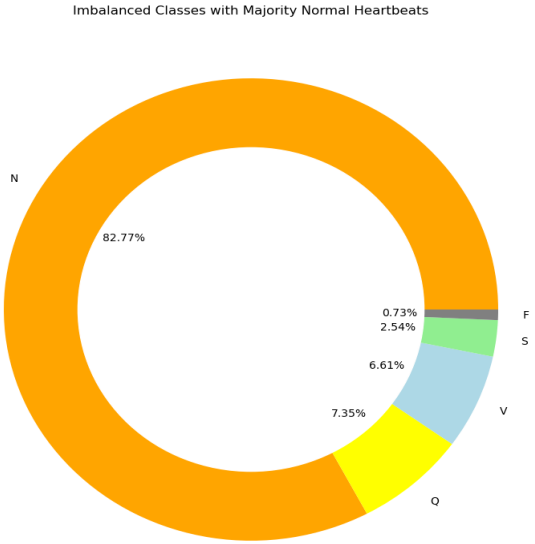


Figure 3. Imbalance Pie-Chart Distribution of Data

Since the minority classes had fewer samples than the majority class, I used upsampling to increase the number of samples in the minority classes. This was done to ensure that the model does not become biased towards the majority class and can accurately classify all classes. The upsampling was performed using the resampling technique, which randomly duplicates samples from the minority classes until they have an equal number of samples as the majority class. After upsampling, all classes had an equal number of samples, resulting in a balanced dataset as it was shown in **Figure 4**. This balanced dataset was used for training and testing the neural network model.

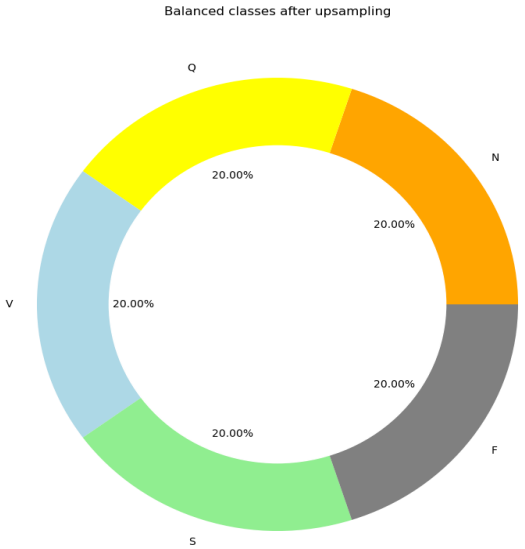


Figure 4. Balance Pie-Chart Distribution of Data

N: This type of heartbeat is a normal sinus rhythm and is considered a healthy heartbeat. It has a consistent pattern with a P wave preceding each QRS complex, indicating a normal electrical pathway through the heart.

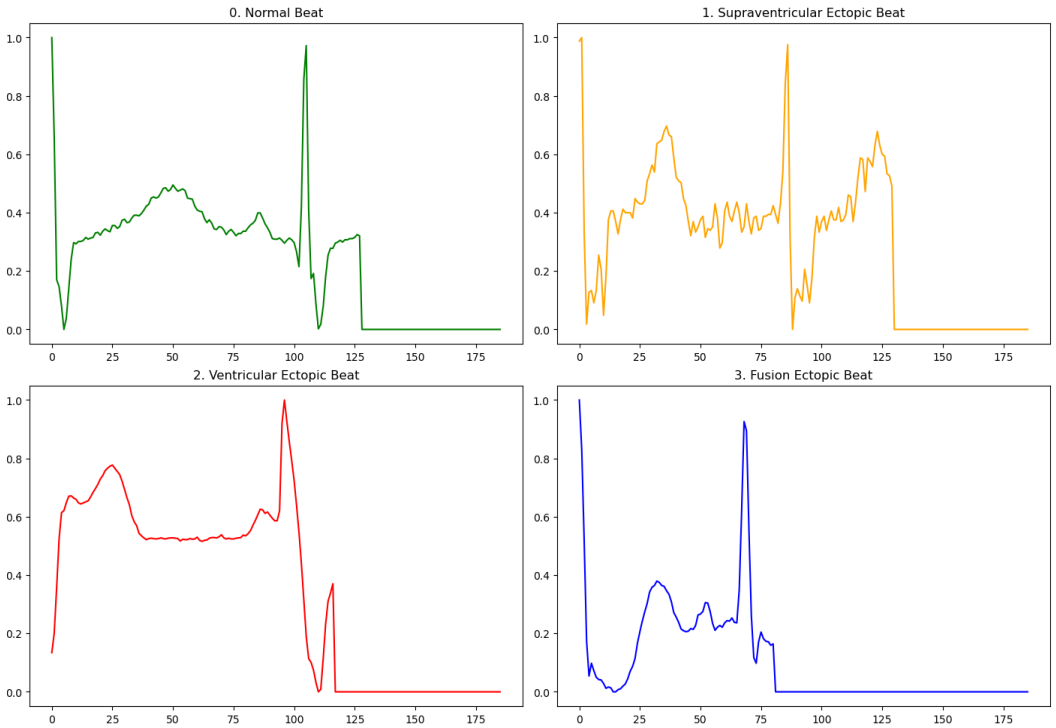
S: This type of heartbeat is called a Supraventricular premature beat, which means it originates from the atria (the upper chambers of the heart) instead of the normal electrical pathway from the sinoatrial (SA) node. It appears earlier than expected, disrupting the normal rhythm of the heart.

V: This type of heartbeat is a Premature Ventricular Contraction (PVC) and originates from the ventricles (the lower chambers of the heart). It is an early beat, which is why it appears differently than a normal beat, with a wider and taller QRS complex.

F: This type of heartbeat is a Fusion of a normal and PVC beat. It occurs when a PVC happens during the normal electrical pathway, leading to a combination of the two patterns in the ECG graph.

Q: This type of heartbeat is a paced rhythm, where an external pacemaker device sends electrical signals to the heart to regulate the heartbeat. This appears differently in the ECG graph than a normal sinus rhythm, with a consistent pattern but different P wave and QRS complex morphology.

Figure 5. will explain the visualization of signals of each type of heartbeat.



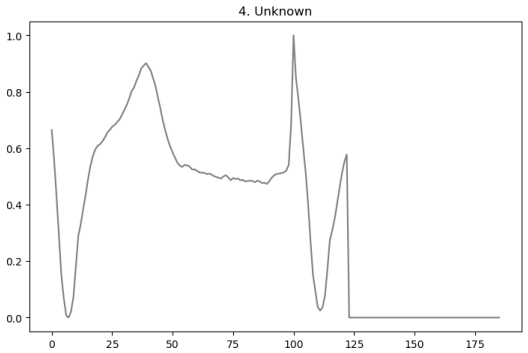


Figure 5. Signal Visualization of each type of Heartbeat

- Modeling

For the modeling stage, I implemented a simple Sequential TFv2 model that utilizes stacked Convolutional layers. The model is capable of summarizing chunks of data based on filter size, which generates feature maps. To classify our data into 5 different classes, I incorporated a softmax output layer. The model consists of three sets of Conv1D layers followed by MaxPooling1D layers, which helps to downsample the data. To prepare the output of the last Conv1D layer for classification, I added a Flatten layer and two Dense layers were added for higher-level feature extraction. Finally, I added a Dense layer with softmax activation to classify the data.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 168, 50)           1050

 max_pooling1d (MaxPooling1D  (None, 168, 5)           0
 )

 conv1d_1 (Conv1D)           (None, 154, 20)           1520

 max_pooling1d_1 (MaxPooling  (None, 154, 1)           0
 1D)

 conv1d_2 (Conv1D)           (None, 145, 10)           110

 max_pooling1d_2 (MaxPooling  (None, 145, 1)           0
 1D)

 flatten (Flatten)           (None, 145)               0

 dense (Dense)               (None, 512)               74752

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 5)                 645

=================================================================
Total params: 143,741
Trainable params: 143,741
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. Signal Visualization of each type of Heartbeat

Figure 6. summarizes the model that was just created. It has a total of 143,741 parameters, and all of them are trainable. The model consists of three Conv1D layers

with 50, 20, and 10 filters respectively, followed by max-pooling layers with pool sizes of 10, 15, and 10. The output of the last max pooling layer is then flattened, and passed through two dense layers with 512 and 128 units respectively, before reaching the final output layer with 5 units and a softmax activation function. The loss function used during training is sparse categorical cross-entropy, and the optimizer used is Adam.

III. Experimental Results

The experimentation results showed that the model achieved a higher accuracy of 96.43% on the unbalanced dataset compared to the accuracy of 93.94% on the balanced dataset. This result suggests that the unbalanced dataset provided more discriminative information to the model, resulting in better classification performance.

```
685/685 - 2s - loss: 0.2831 - acc: 0.9394 - 2s/epoch - 3ms/step
Accuracy for the Balanced Dataset =  93.94 %

685/685 - 2s - loss: 0.2031 - acc: 0.9643 - 2s/epoch - 3ms/step
Accuracy for the Unbalanced Dataset =  96.43 %
```

Figure 7. Comparison of Balanced and Unbalanced Dataset

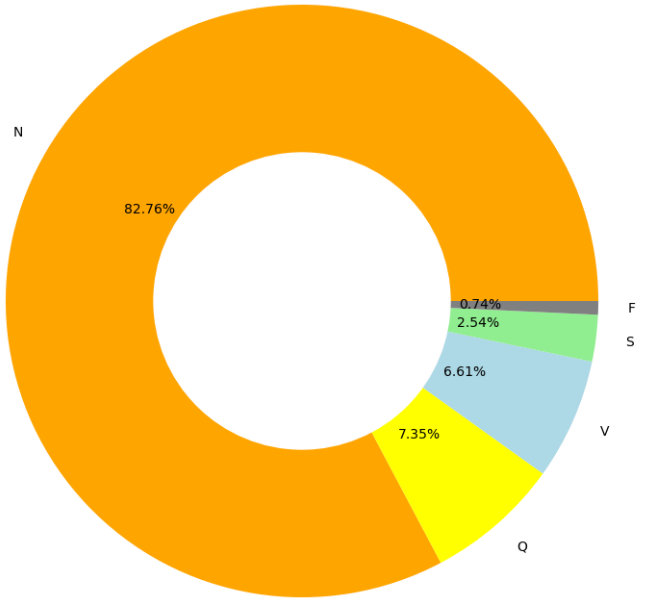


Figure 8. Pie-Chart Visualization of the Results

The reason behind the higher accuracy of the unbalanced dataset is that the model might have been overfitting to the minority classes when using the balanced dataset. By upsampling the minority classes to balance the dataset, we increased the representation of these classes in the training set, but at the same time, we introduced duplicated data points, which might have caused the model to memorize these data points rather than learn the general patterns in the data.

IV. Discussion

In contrast, the unbalanced dataset has a more natural distribution of classes, which might have helped the model to learn better and generalize to unseen data. However, it's important to note that this result might not generalize to other datasets, and the best approach to handling imbalanced data might depend on the specifics of the problem at hand.

Additionally, the model may overfit the balanced dataset during training, leading to lower performance on the validation set. This could be because the balanced dataset has a relatively small sample size compared to the unbalanced dataset, so the model may have been unable to learn the full complexity of the data. Furthermore, it's possible that the model architecture and hyperparameters were not optimized for the balanced dataset, as it has a different class distribution and statistical properties than the original unbalanced dataset.

The discrepancy in validation scores between the balanced and unbalanced datasets highlights the importance of carefully considering the class distribution and statistical properties of the data when designing and training machine learning models. Additionally, it underscores the need for thorough validation procedures, including cross-validation and evaluation on multiple datasets, to ensure that models generalize well to new and unseen data.

Based on the achieved results, it can be concluded that the deep learning model implemented in this project performed well in accurately classifying the ECG heartbeats into the five classes. The model achieved a high accuracy of 96.43% on the unbalanced dataset and 93.94% on the balanced dataset. However, it is important to note that the model was trained on a relatively small dataset and there may be limitations to its generalizability to larger datasets. Additionally, further exploration and analysis could be done to optimize the model's performance, such as experimenting with different hyperparameters or incorporating additional features. Overall, the results are promising and suggest that deep learning models could be useful tools in ECG heartbeat classification for medical diagnosis purposes.

Appendix: Source Code

Upsampling to Ensure an Even Class-Distribution

As we can observe in the above class-wise data distribution, 'N' has the highest number of records, with very few records for 'F' class.

In [12]:
```
df0=mitbih_train[mitbih_train[187]==0].sample(n=20000,random_state=10)
df1=mitbih_train[mitbih_train[187]==1]
df2=mitbih_train[mitbih_train[187]==2]
df3=mitbih_train[mitbih_train[187]==3]
df4=mitbih_train[mitbih_train[187]==4]

df1_upsampled=resample(df1,replace=True,n_samples=20000,random_state=100)
df2_upsampled=resample(df2,replace=True,n_samples=20000,random_state=101)
df3_upsampled=resample(df3,replace=True,n_samples=20000,random_state=102)
df4_upsampled=resample(df4,replace=True,n_samples=20000,random_state=103)
train_df=pd.concat([df1_upsampled,df2_upsampled,df3_upsampled,df4_upsampled,df0])

print(train_df[187].value_counts())
plt.figure(figsize=(10,10))
plt.pie(train_df[187].value_counts(), labels=['N','Q','V','S','F'],
        colors=['orange','yellow','lightblue','lightgreen','grey'], autopct='%.2f%%')
plt.gcf().gca().add_artist(plt.Circle( (0,0), 0.7, color='white' ))
plt.title('Balanced classes after upsampling')
plt.show()
```

Modelling

Creating the Model by Stacking Up Conv1D Layers

We will implement a simple Sequential TFv2 model with stacked up Convolutional layers.

These will essentially summarize chunks of data together based on filter-size, to produce feature-maps.

Finally we'll have a softmax output layer to give 5 predictictions since we have 5 classes.

In [13]:
```
model=tf.keras.Sequential()

model.add(tf.keras.layers.Convolution1D(filters=50,kernel_size=20,activation='relu',kernel_initializer='glorot_uniform'

#a1_0=> 187-20+1= 168,50
model.add(tf.keras.layers.MaxPool1D(pool_size=10,data_format='channels_first'))

#a1_1=> 50//10= 168,5
model.add(tf.keras.layers.Convolution1D(filters=20,kernel_size=15,activation='relu',kernel_initializer='glorot_uniform'

#a2_0=> 168-15+1= 154,20
model.add(tf.keras.layers.MaxPool1D(pool_size=15,data_format='channels_first'))

#a2_1=> 20//15= 154,1
model.add(tf.keras.layers.Convolution1D(filters=10,kernel_size=10,activation='relu',kernel_initializer='glorot_uniform'

#a3_0=> 154-10+1=145,10
model.add(tf.keras.layers.MaxPool1D(pool_size=10,data_format='channels_first'))

#a3_1=> 10//10=145,1
model.add(tf.keras.layers.Flatten())

#a4=> 145
model.add(tf.keras.layers.Dense(units=512,activation='relu',kernel_initializer='glorot_uniform'))

#a4=> 512
model.add(tf.keras.layers.Dense(units=128,activation='relu',kernel_initializer='glorot_uniform'))

#a5=> 128
model.add(tf.keras.layers.Dense(units=5,activation='softmax'))

model.compile(optimizer='Adam',loss='sparse_categorical_crossentropy',metrics=['acc'])
model.summary()
```

Training the Model Part I

Balanced Dataset

In [14]:
```
mitbih_test_x2=np.asarray(mitbih_test_x)
mitbih_test_x2=mitbih_test_x2.reshape(-1,187,1)
```

Dataset after Up-Sampling with an Even Distribution across the Classes

In [15]:
```
train_df_X=np.asarray(train_df.iloc[:,:187]).reshape(-1,187,1)
train_df_Y=train_df.iloc[:,187]
print(train_df_X.shape)

hist=model.fit(train_df_X,train_df_Y,batch_size=64,epochs=20)
```

Training the Model Part II

Unbalanced Dataset

In [18]:
```
full_train_x2=np.asarray(full_train_x)
full_train_x2=full_train_x2.reshape(-1,187,1)
print(full_train_x2.shape)

hist=model.fit(full_train_x2,full_train_y,batch_size=64,epochs=20)
```

References

F Shayan, K Mohammad, and S Majid, 2018 IEEE International Conference on Healthcare Informatics. *ECG Heartbeat Classification: A Deep Transferable Representation*. Available at: https://arxiv.org/pdf/1805.00794.pdf (Accessed: March 15, 2023)

A Saira, A Sajid, and A Mohamed-Slim, 2021, Sci Rep 11, 18738. *ECG-based machine-learning algorithms for heartbeat classification*. Available at: https://www.nature.com/articles/s41598-021-97118-5#citeas (Accessed: March 16, 2023)

Attachments

Dataset Used
https://www.kaggle.com/datasets/shayanfazeli/heartbeat

Google Colaboratory Source Code
https://colab.research.google.com/drive/12EmjI-g4OM9SPN5rgQGfsyObscXR8iVN