Development of a Bagging-based Ensemble Model for ECG Classification

by

Hammada Ballali Ebbi

Supervisor: Tim Bashford

Project submitted as part of the requirements for the award of Master of Science Software Engineering and Artificial Intelligence

August 2024

AKNOLWEDGEMENT

I would like to express my heartfelt gratitude to my dissertation supervisor, Dr. Tim Bashford, for his invaluable guidance and support throughout this research project. His deep expertise, insightful feedback, and constructive criticism have been crucial in shaping the direction of my work and improving its quality.

I also extend my thanks to the teachers at The Wales Institute of Science and Art, who have guided me through these years of learning, and to the staff at the University of Wales Trinity Saint David for providing the essential resources and support needed to complete this dissertation. The stimulating academic environment and rigorous coursework have thoroughly prepared me for this research.

Lastly, I am deeply appreciative of my family and friends for their unwavering support and encouragement throughout my academic journey. Their love, understanding, and patience have been a constant source of inspiration and motivation.

Thank you all for your continued support and encouragement.

Contents

1. Introduction 1
1.1 Aims and Objectives 4
2. Related Work 4
This section examines the latest advancements in the development of Deep Learning models for ECG classification. The goal is to identify the most effective the most efficient Deep Learning models to be used in designing a bagging-based ensemble model. 4
3. Methodology 8
3.1 Dataset processing 9
3.1.1 PTBDB dataset for binary classification 9
3.1.2 MITBIH dataset for multiclass classification 13
4.2 Base Deep Learning models 20
4.2.1 CNN model 20
4.2.2 LSTM model 21
4.2.3 Transformer-based model 22
3.2 Bagging technique implementation 22
4. Design and Implementation 23
5. Results and Discussion 24
5.1 Results of binary classification on PTB dataset 24
5.2 Results of multiclass classification on MITBIH dataset 27
6. Conclusions and Recommendations (Future work) 30
7. Reflection (Personal Reflections on the Projects) 31
8. References 32

Figures

Figure 1. P, QRS and T waves by Barhatte et al. [12] 2
Figure 2. Trend of different ECG Datasets used in the recent years for Deep Learning ECG classification research by Xiao et al. [19]. 2
Figure 3. Initial balance of the PTBDB dataset 10
Figure 4. Balanced dataset. 11
Figure 5. Normal ECG signal from PTBDB 12
Figure 6. Abnormal ECG signal from PTBDB 12
Figure 7. Initial balance of the MITBIH training dataset 15
Figure 8. Balanced MITBIH training dataset bar chart. 16
Figure 9. Normal ECG signal from MITBIH training dataset 17
Figure 10. SEB ECG signal from MITBIH training dataset. 18
Figure 11. VEB ECG signal from MITBIH ECG dataset. 18
Figure 12. FB ECG signal from MITBIH training dataset. 19
Figure 13. Unknown ECG signal from MITBIH training dataset. 19

Tables

Table 1. CNN for binary classification over 10 epochs. 24
Table 2. LSTM for binary classification over 10 epochs. 25
Table 3. Transformer-based model for binary classification over 10 epochs. 26
Table 4. Metrics for binary ECG classification on PTB dataset 27
Table 5. CNN for multiclass classification over 10 epochs. 27
Table 6. LSTM for multiclass classification over 10 epochs. 28
Table 7. Transformer-based model for multiclass classification over 10 epochs. 29
Table 8. Metrics for binary ECG classification on MITBIH dataset 29

Abstract

The importance of computational methods, particularly the application of machine learning models in cardiovascular disease classification and recognition, is rapidly growing. CNN, LSTM, and Transformer models have demonstrated in various studies that, when implemented with robust architectures and supported by ample datasets, they can achieve highly accurate results. This study explores the application of bagging techniques to three base models: CNN, LSTM, and Transformer, for both binary classification on the PTB dataset and multiclass classification on the MITBIH dataset. The findings indicate that the CNN model outperforms the other two models under the selected parameters and across ten epochs achieving 0.95 for binary classification and 0.96 for multiclass classification. Additionally, the use of bagging techniques results in a slight deterioration, likely due to the weaker performance of the Transformer and LSTM models.

Keywords:

Cardiovascular Disease (CDV)

Deep Learning (DL)

Electrocardiogram (ECG)

Normal Sinus Rhythm (NSR)

Supraventricular Ectopic Beats (SEB)

Ventricular Ectopic Beats (VEB)

Fusion Beat (FB)

Convolutional Neural Network (CNN)

Long Short-Term Memory (LSTM)

1. Introduction

According to World Health Organisation (WHO) reports [1], Cardiovascular Diseases (CVDs) are still among the leading causes of death globally. In 2019, an estimated 17.9 million people died from CVD, representing 32% of all deaths worldwide. CVDs encompass a range of conditions, including coronary artery disease [2], hypertension [3] and heart failure [4], all of which can have devastating impacts on individuals and communities. Among these conditions, arrhythmias [5], which are irregular heartbeats, are particularly concerning. Arrhythmias can lead to serious complications such as stroke, heart failure, or sudden cardiac arrest if not properly managed. This emphasizes the need for comprehensive strategies to prevent and manage CVDs, including the detection and treatment of arrhythmias. Advancements in medical technology and treatment options, such as implantable devices like pacemakers [6] and defibrillators , play a significant role in managing arrhythmias effectively . By focusing on the prevention and early detection of arrhythmias, alongside other cardiovascular conditions, it is possible to reduce the burden of CVDs and improve overall public health outcomes.

The electrocardiograph, first invented by Willem Einthoven in 1902 [7] , emerged as a powerful tool for the diagnosis of different CVDs. An electrocardiogram (ECG) is a test that records the heart's electrical activity by placing electrodes on the skin to detect the impulses generated during the heart's contraction and relaxation phases [8], [9]. These impulses are graphically displayed as waves, each representing a different phase of the cardiac cycle. Clinicians utilize ECGs to diagnose various heart conditions, including arrhythmias, myocardial infarctions (heart attacks), and structural or functional abnormalities of the heart. Interpreting an ECG requires specialized training due to the nuanced patterns and subtle waveform changes that can indicate underlying cardiac issues, making it an essential tool in cardiology for diagnosing and monitoring heart health.

Typically, a standard 12-lead ECG [10], [11] is employed as a diagnostic tool to evaluate the heart's electrical activity from multiple perspectives. It features 12 distinct views, or "leads," that offer unique angles of the heart's electrical signals. These leads are obtained by placing electrodes on specific body locations, enabling a thorough assessment of heart function. The 12 leads include six limb leads (I, II, III, aVR, aVL, aVF) and six precordial leads (V1-V6), each providing information about different regions of the heart's electrical activity. By analysing the patterns and variations in the electrical signals across these leads, healthcare professionals can detect abnormalities such as ischemia, infarction, arrhythmias, and conduction disorders with enhanced accuracy and specificity, facilitating the diagnosis and management of cardiac conditions.

The main purpose of ECG classification is to diagnose diseases by analysing ECG waves, especially the P wave, QRS complex and T wave [12]. These waves provide valuable information about the presence and type of diseases by different diagnostic methods (Fig 1).

Figure 1. P, QRS and T waves by Barhatte et al. [12]

Over the past few decades, the computerized identification of ECGs has emerged as a widely adopted practice, aiding cardiologists in categorizing extended ECG recordings.

Deep learning, leveraging its robust feature extraction capabilities, has achieved remarkable accuracy in the classification of ECG signals, serving as a potent computer-aided method in this domain [9], [13], [14].

One of the primary challenges in developing and implementing machine learning models for ECG classification is the limited availability of dataset sources, primarily due to the sensitive nature of the data. However, there are some publicly available datasets that are widely used in research of this type. One of the most widely used and recognized datasets in the research is the MIT-BIH dataset [15]. This dataset consists of 23 randomly chosen recordings and 25 recordings selected for less common but significant arrhythmias, collected between 1975 and 1979. The recordings, digitized at 360 samples per second with 11-bit resolution, were annotated by multiple cardiologists. The database, containing approximately 110,000 annotations, has been freely available on PhysioNet since September 1999, with additional files posted in February 2005.

Another widely used dataset in the field of ECG research is the PTB-XL dataset [16]. It was gathered over nearly seven years using devices from Schiller (1989 - 1996). The ECG samples were initially recorded as part of an extensive project of the Physikalisch-Technische Bundesanstalt (PTB). The data remained restricted to the public until late 2019. When it was decided to release the records to the public, some adjustments were made to facilitate usability and accessibility to a wider public and to enrich research in the field.

Although there are some other publicly available datasets such as INCART 12- lead Arrhythmia Database [17], or the Fantasia Database [18], a study by Xio et al. in 2023 [19] that compares the use of different datasets for ECG classification (Fig 2) finds that the use of these two databases is the most widespread, mainly due to their extensive and exhaustive documentation and the large number and variety of samples of which they are composed, allowing the classification of a wide range of conditions, which makes them very valuable for ECG research.

Figure 2. Trend of different ECG Datasets used in the recent years for Deep Learning ECG classification research by Xiao et al. [19].

In this study, the two most relevant datasets are utilized to assess a bagging-based ensemble model. The PTB dataset was used to evaluate binary classification performance, while the multiclass MITDB dataset was employed to assess multiclass classification performance. The TPB dataset provides to classes: Normal and Abnormal ECGs. The MITBIH dataset provides five

classes: Normal Sinus Rhythm (NSR), Supraventricular Ectopic Beat (SEB), Ventricular Ectopic Beat (VEB), Fusion Beat (FB), and a class for the unknown beats (Q).

A normal sinus rhythm (NSR) is the heart's regular rhythm, initiated by the sinoatrial (SA) node, with a heart rate between 60 and 100 beats per minute [20]. It features consistent time intervals between beats, with each heartbeat preceded by a normal P wave, a PR interval of 0.12 to 0.20 seconds, and a narrow QRS complex lasting less than 0.12 seconds. The T waves are appropriately upright in specific leads, the ST segment is flat and aligns with the baseline, and the QT interval is appropriately adjusted for the heart rate. This rhythm indicates healthy, regular electrical activity within the heart.

Supraventricular ectopic beats are premature heartbeats originating above the ventricles, typically in the atria or the atrioventricular node. These beats are often benign and can occur in healthy individuals, but they can also be a sign of underlying cardiac conditions. They may cause the heart to beat irregularly or prematurely [21]. On an ECG, these beats are identified by examining the timing and morphology of the P-wave and the RR intervals. SVBs often have a shorter RR interval and an abnormal P-wave but a normal QRS complex.

Ventricular ectopic beats are premature heartbeats that originate from the ventricles, the lower chambers of the heart. These beats are more concerning than supraventricular ectopic beats because they can be associated with more serious cardiac conditions, including ventricular tachycardia or ventricular fibrillation, which can be life-threatening if not treated. [22]. VEBs typically have a wide and bizarre-looking QRS complex without a preceding P-wave. They often occur earlier than expected and are followed by a compensatory pause.

A fusion beat occurs when a normal heartbeat and an ectopic beat (either supraventricular or ventricular) coincide. The resultant ECG waveform is a combination of the two signals. Fusion beats are important to identify because they can indicate the presence of competing pacemaker activity in the heart [23]. A fusion beat is identified by detecting a QRS complex that has a combination of features from both normal and ectopic beats, indicating simultaneous activation of the ventricles by both normal and ectopic pacemakers. The 'Q' class in the MITBIH dataset is used for heartbeats that do not fit into the other categories and are not well-defined or are of unknown origin. This class might include noisy signals, artifact-induced patterns, or unclassifiable beats that the algorithm or annotator could not definitively categorize.

The design of deep learning (DL) models plays a critical role in the field of computing-based ECG classification. These models typically feature multi-level or multi-layer architectures, with each level or layer acting as a feature extractor that progressively refines the representation of signal characteristics. Depending on the key feature extractors within the neural networks, the DL classification models examined in the selected studies can be primarily categorized into several types: convolutional neural networks (CNNs) [24], [25], [26], [27], [28], long short-term memory (LSTM) [29], [30], [31] and transformers [32], [33], [34], in addition to some studies some that have investigated the potential of combining the outputs of different models through ensemble learning techniques [35], [36], [37].

1.1 Aims and Objectives

The aim of this research is to develop a highly accurate deep learning model based on bagging techniques for 12-lead ECG classification, specifically targeting four types of rhythms, to improve diagnostic efficiency and accessibility in cardiac health monitoring. The underlying hypothesis is that ECG classification can be enhanced by implementing a bagging-based ensemble approach applied to various deep learning models.

To achieve this goal, the following objectives have been outlined:

- **Explore current ECG classification methods**: Conduct a comprehensive literature review to understand recent advancements in ECG classification, particularly those using ensemble techniques in deep learning.
- **Develop a bagging-based deep learning model**: Create a model that combines different deep learning approaches to achieve accurate ECG classification.
- **Evaluate model performance across diverse rhythms**: Test the model's performance on different heart rhythms to ensure its robustness and generalization capability. This will involve evaluating the model's performance on the PTB dataset for binary ECG classification and on the MITBIH for multiclass ECG classification.

2. Related Work

This section examines the latest advancements in the development of Deep Learning models for ECG classification. The goal is to identify the most effective the most efficient Deep Learning models to be used in designing a bagging-based ensemble model.

In recent years, many studies have explored several approaches for designing and developing deep learning models for ECG classification.

A commonly explored approach for ECG classification tasks is the implementation of CNN models.

A study by Mi et al. [24], focuses on preprocessing techniques for ECG data, particularly highlighting the use of the MIT-BIH Arrhythmia Database. It details the implementation of wavelet transform methods to enhance ECG signal accuracy, which is crucial for effective classification. The improved signal maps derived from this process are used to train a Back Propagation (BP) neural network, which is adept at recognizing and classifying various types of arrhythmias. These preprocessing steps are vital in ensuring that the data fed into machine

learning models is of high quality, thereby improving the models' performance in detecting and classifying heart conditions accurately.

A study by Ekİnci et al. [25], explores the impact of various preprocessing techniques on the performance of deep learning models for ECG classification. It emphasizes filtering out Baseline Wander (BW) and Powerline Interference (PLI) noise. The study compares Short-time Fourier Transform (STFT) and Continuous Wavelet Transform (CWT) for feature extraction, finding that BW noise significantly affects classification accuracy. Using a low-complexity CNN with less than 90k parameters, the optimal preprocessing achieved an average F1-Score of 90.11%.

A research by Śmigiel et al. [26], explores preprocessing techniques like filtering out baseline wander and powerline interference, and feature extraction using wavelet transform and entropy-based features. Three deep learning models were used: a convolutional network, SincNet, and a convolutional network with entropy-based features. The convolutional network with entropy features achieved the highest classification performance with an F1-Score of 90.11%, while the standard convolutional network offered the best computational efficiency.

In another 2024 publication by Z. Chen et al. [27], introduces a novel hybrid method to address imbalanced datasets by employing amplitude adjustment and weighted loss functions during model training. The study also enhances depth wise and pointwise convolutions into one-dimensional operations specifically tailored for ECG heartbeat signals, which reduces model complexity while maintaining accuracy, making it suitable for resource-constrained wearable devices. By combining imbalanced dataset mitigation techniques with hardware-efficient deep learning networks for arrhythmia classification, this approach improves dataset balancing and data compression. Compared to existing methods, it enhances performance while significantly reducing trainable parameters and computations in traditional CNN models.

Another research by S. Oh and M. Lee [28] proposes a shallow domain knowledge injection (SDK-Injection) method to improve CNN-based ECG pattern classification. Pre-processing techniques involve the application of a Savitzky-Golay filter for noise reduction through curve fitting-based smoothing. The pre-processed data is then subjected to SDK-injecting attention to embed shallow domain knowledge, focusing on important sub-patterns like the T wave, which are crucial for diagnosing conditions such as ischemia. The final step involves expanding the univariate ECG data into a multivariate format by adding variants, including the smoothed data, attention output, and smoothed attention output. This enriched input is fed into a CNN, enhancing its performance. The model was trained and tested on multiple ECG datasets, demonstrating significant improvements in accuracy, precision, sensitivity, specificity, and F1-score, especially in balanced datasets. The proposed method effectively leverages shallow domain knowledge to boost the classification accuracy of existing parameter-optimized CNN models for ECG analysis.

Another important avenue of research in recent years is the application of LSTM models.

A study by L. D. Sharma et al (2023) [29] , presents a method for classifying cardiac arrhythmias using ECG signals. The data processing techniques involve the use of Stationary Wavelet Transform (SWT) to preprocess the raw ECG signals, effectively decomposing them into various

frequency sub-bands to remove noise and enhance the signal. The SWT is particularly beneficial because it preserves the length of the signal throughout the decomposition process, which is crucial for accurate reconstruction and analysis. Then, a Bi-directional Long Short-Term Memory (Bi-LSTM) network, which is compared against Recurrent Neural Networks (RNN) and Gated Recurrent Units (GRU). The Bi-LSTM model showed superior performance, achieving an overall accuracy of 99.72%, with high precision, sensitivity, and specificity across all classes of arrhythmias considered in the study. This performance significantly outperforms the RNN and GRU models, making the Bi-LSTM approach highly effective for the task of ECG-based arrhythmia classification.

Another research by M. Karri and C. S. R. Annavarapu (2023) [30], describes a real-time embedded system designed for QRS-complex detection and arrhythmia classification using Long Short-Term Memory (LSTM) networks, leveraging hybridized features. The data processing techniques include the use of Discrete Wavelet Transform (DWT) and Delta Sigma Modulation (DSM) for feature extraction, focusing on QRS detection and wavelet-based noise reduction. The ECG signals are preprocessed to remove noise, and critical features such as R peak, onset, and offset of P and T waves are extracted. These hybrid features are then fed into an LSTM model for arrhythmia classification. The system was tested using the MIT-BIH arrhythmia database, achieving impressive results with an accuracy of 99.64%, sensitivity of 99.87%, positive predictivity of 99.15%, and an F1 score of 98.18%. These metrics demonstrate the high effectiveness of the proposed system in accurately classifying arrhythmias in real-time on a low-power embedded device.

A study by S. Boda et al. (2023) [31] introduces an automated patient-specific ECG beat classification system using Long Short-Term Memory (LSTM)-based recurrent neural networks (RNNs). The data processing techniques include pre-processing the ECG signals to remove noise, followed by QRS complex detection using the Pan-Tompkins algorithm. The ECG signals are then segmented into individual beats, and temporal and morphological features are extracted. These features are combined with the LSTM network to capture the temporal dependencies in the ECG waveform for arrhythmia classification. The system was evaluated using the MIT-BIH arrhythmia database, achieving superior performance with an accuracy of 99.64%, sensitivity of 99.87%, specificity of 99.72%, and an F1 score of 98.36%. These metrics demonstrate the effectiveness of the LSTM-based approach in accurately classifying various types of arrhythmias, making it suitable for real-time, patient-specific ECG analysis in clinical settings.

In addition to the above, several recent studies have focused on the implementation of transformer-based models.

A study by H. El-Ghaish and E. Eldele (2024) [32], introduces "ECGTransForm," a deep learning framework designed for ECG arrhythmia classification, incorporating advanced data processing techniques and a novel machine learning model. The data processing involves segmenting ECG signals into discrete windows, normalizing the amplitude, detecting R-peak candidates, and segmenting signals into fixed lengths using zero-padding. The proposed model, ECGTransForm, leverages Multi-scale Convolutions to capture spatial features at various scales, a Channel Recalibration Module to refine feature representation by considering interdependencies across

channels, and a Bidirectional Transformer (BiTrans) to capture temporal dependencies from both past and future contexts. Additionally, the model uses a Context-Aware Loss function to address class imbalance by dynamically adjusting class weights. The model was evaluated on the MIT-BIH and PTB Diagnostic ECG databases, achieving impressive results, including an accuracy of 99.35% and a macro-average F1-score of 94.26% on the MIT-BIH dataset, outperforming several state-of-the-art methods.

In 2023, Y. Dong et al. [33] presented an arrhythmia classification model based on a Vision Transformer (ViT) with deformable attention, termed CNN-DVIT. The data processing includes the use of spatial pyramid pooling to handle varied-length ECG signals, allowing the model to process inputs of different sizes effectively. The machine learning model combines convolutional neural networks (CNNs) for feature extraction and a Vision Transformer with deformable attention for robust classification. The model achieved an F1 score of 82.9% on the CPSC-2018 dataset, outperforming other transformer-based models in ECG arrhythmia classification.

Another study by A. Varghese et al. (2023) [34], presents a method for classifying ECG arrhythmias using a transformer-based model, specifically DistilBERT. The data processing includes denoising ECG signals with Butterworth filters and segmenting them around the R peak. The Synthetic Minority Oversampling Technique (SMOTE) is used to balance the dataset. The model, which omits the input embedding step, achieved remarkable results on the MIT-BIH dataset with an accuracy of 99.92%, and precision, recall, and F1-score all at 0.99, along with a ROC-AUC score of 0.999. These metrics demonstrate the model's high performance in classifying various arrhythmias.

As far as ensemble models are concerned, recent studies have put forward different approaches.

In a 2024 paper by Morteza Maleki [35], a model utilizing Wavelet transformation is presented for feature extraction from ECG signals. This approach enhances the classification accuracy of various cardiovascular diseases using machine learning techniques by capturing both time and frequency information, which aligns well with the complex nature of ECG signals. The study underscores the effectiveness of wavelet-based feature extraction, emphasizing the importance of selecting appropriate wavelets and optimizing feature extraction depth for better classifier accuracy. Notably, Random Forest and Gradient Boost classifiers showed superior performance, demonstrating their ability to handle complex patterns and large feature sets derived from wavelet transformations. This advancement promises to enhance the reliability of automatic ECG analysis systems in clinical settings.

A recent study by W. Ji and D. Zhu [36] presents a new method by integrating Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU) at the initial signal processing stage for analysing simulated signals. Initially, the signal undergoes processing with a Butterworth high-pass filter to minimise power line noise interference, followed by wavelet transform to reduce electromyographic interference. One-dimensional CNN is employed for automatic feature extraction, while the fusion of GRU with CNN addresses the limited time dependency inherent in CNN networks for ECG signals. Acknowledging potential long-term dependencies in ECG

signals, recurrent neural networks such as GRU can capture these, enhancing classification accuracy and bolstering the network's resistance to noise. Consequently, the model gains improved insight into and captures temporal relationships within different segments of the ECG signal, thereby augmenting classification accuracy.

The study by Plawiak and Acharya (2020) [37] introduces an ECG classification system employing various preprocessing methods and genetic algorithms for optimization. Preprocessing involves gain reduction, constant component reduction, and three normalization techniques: standardization, rescaling, and no normalization. Evaluation metrics include accuracy, specificity, sensitivity, false positive rate, positive predictive value, Fleiss' Kappa, optimization time, training time, classification time, and the acceptance feature coefficient. The results demonstrate that normalization and preprocessing significantly impact classifier performance, with Random Forest and Gradient Boost classifiers showing superior performance in handling complex patterns and large feature sets derived from these preprocessing methods, promising advancements in automatic ECG analysis systems' reliability in clinical settings.

The analysis of various studies reveals that commonly used models like CNN, LSTM, and Transformer consistently achieve high accuracy and efficiency in ECG classification. Some studies have also explored ensemble learning by combining multiple models, resulting in enhanced performance. This study, therefore, proposes a model that applies bagging techniques to these three prevalent models in ECG classification to assess whether their combined application can lead to improved outcomes.

3. Methodology

As highlighted in the literature review, the most effective ECG classification results are achieved using three types of deep learning models: CNN models, LSTM models, and transformer-based models. However, the results obtained leave room for improvement. Therefore, this study

focuses on the development of a model that applies bagging techniques combining predictions from the three types of models.

To evaluate the model's efficiency in different environments, the experiment first conducts binary classification on the PTBDB dataset, followed by multiclass classification on the MITBIH dataset.

3.1 Dataset processing

The experiment is conducted on two distinct datasets to evaluate the efficiency of binary and multiclass classification.

3.1.1 PTBDB dataset for binary classification

The PTB Diagnostic ECG Database, hosted on PhysioNet and curated by the Physikalisch-Technische Bundesanstalt, includes 549 high-resolution 15-lead ECG recordings from 290 subjects, ranging in age from 17 to 87. These subjects include both healthy individuals and patients with various heart conditions, such as myocardial infarction, cardiomyopathy, and arrhythmias. The ECGs are digitized at 1000 samples per second with detailed clinical summaries available for most records. This dataset is primarily used for research, algorithmic benchmarking, and teaching in the field of cardiology. It is available on Kaggle [38] as two separate CSV files: one with normal ECG samples and the other with abnormal ECG samples.

The first step is to merge both csv files in one single data frame using:

full_ptb=pd.concat([ptb_normal,ptb_abnormal],axis=0)

The resulted dataset consist in __ rows and 188 columns.

The resulting data frame must then be randomly shuffled and further divided into training and test data sets:

full_ptb_shuffled = full_ptb.sample(frac=1).reset_index(drop=True)

The last row (index 187) holds the classes 0 for normal ECG and 1 for abnormal ECG.

The initial number of classes is imbalanced, containing 10506 abnormal samples and 4046 normal samples:

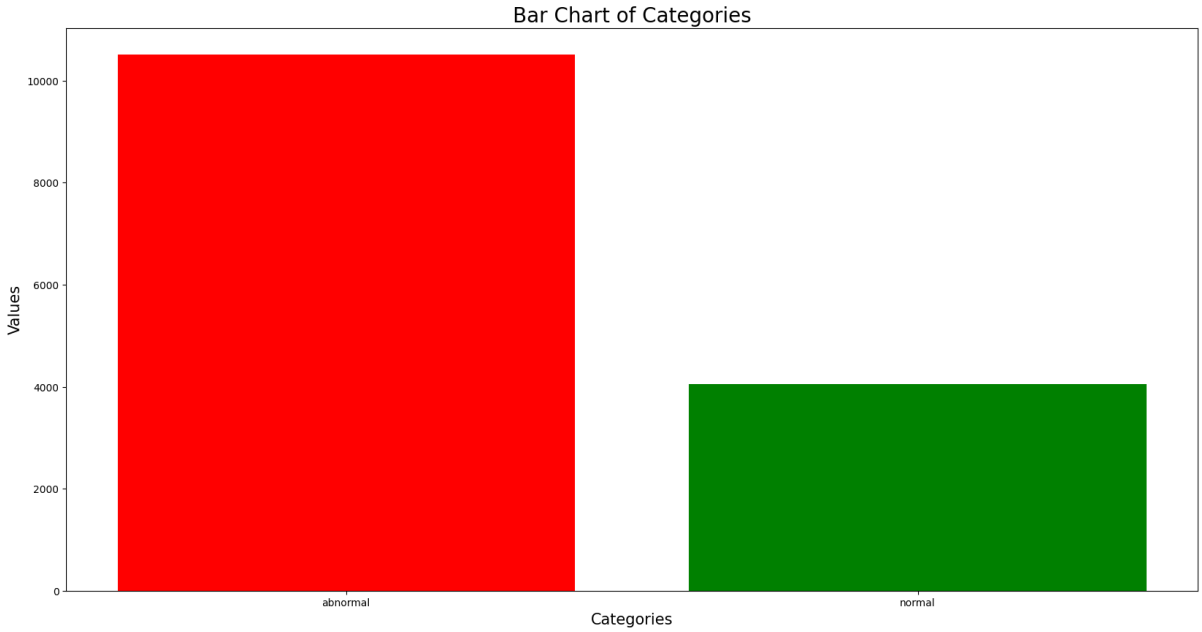

Figure 3. Initial balance of the PTBDB dataset

Therefore, the number of abnormal samples must be reduced to fit the number of normal samples:

```
df_1=(full_ptb_shuffled[full_ptb_shuffled[187]==0]).sample(n=4046,random_state=42)

df_2=(full_ptb_shuffled[full_ptb_shuffled[187]==1]).sample(n=4046,random_state=42)

full_ptb_shuffled_balanced=pd.concat([df_1,df_2])
```

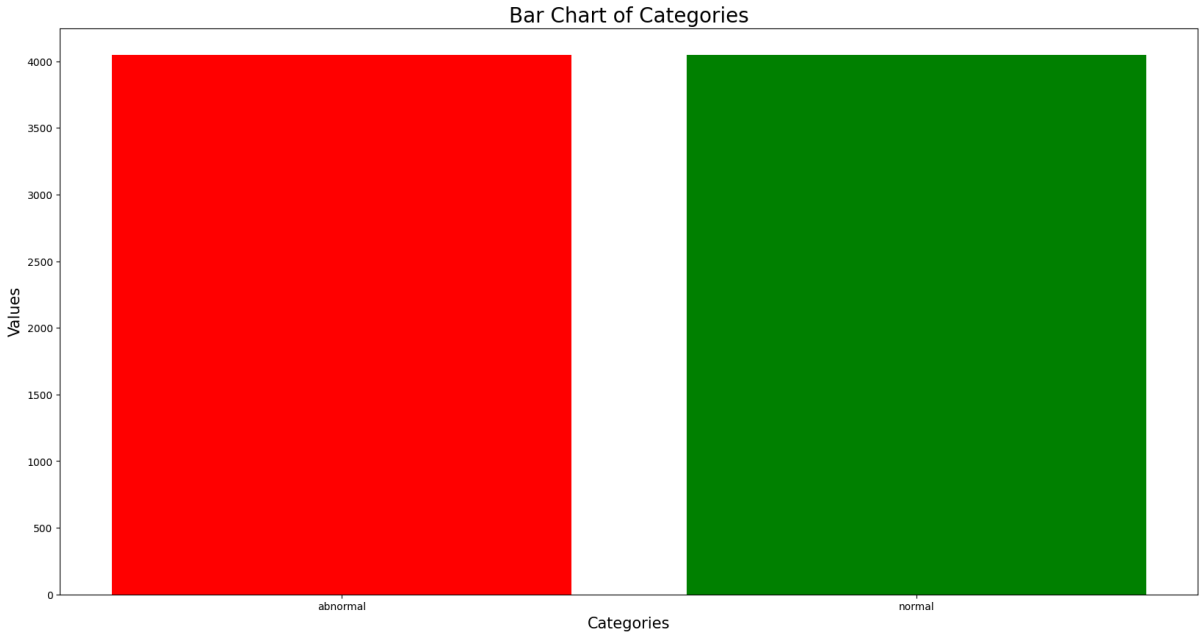

Figure 4. Balanced dataset.

We then plot an ECG sample of each category using the plot library tools:

```
row = 4

# Extract the signal from the chosen row

ECG_signal = full_ptb_shuffled_balanced.iloc[row, : -1]

# Plot the signal

plt.figure(figsize=(15, 5))

plt.plot(ECG_signal)

plt.title(f"Signal from row {row}")

plt.show()
```

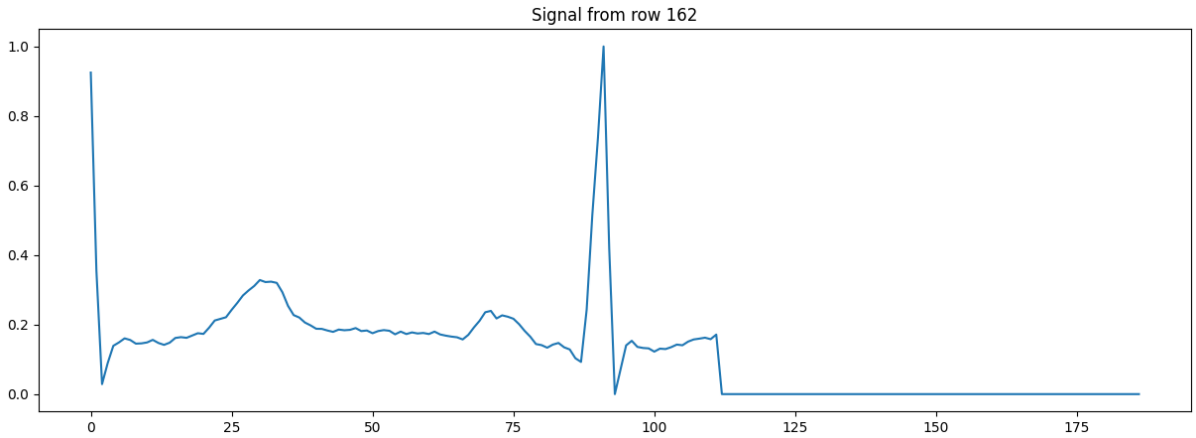

Figure 5. Normal ECG signal from PTBDB

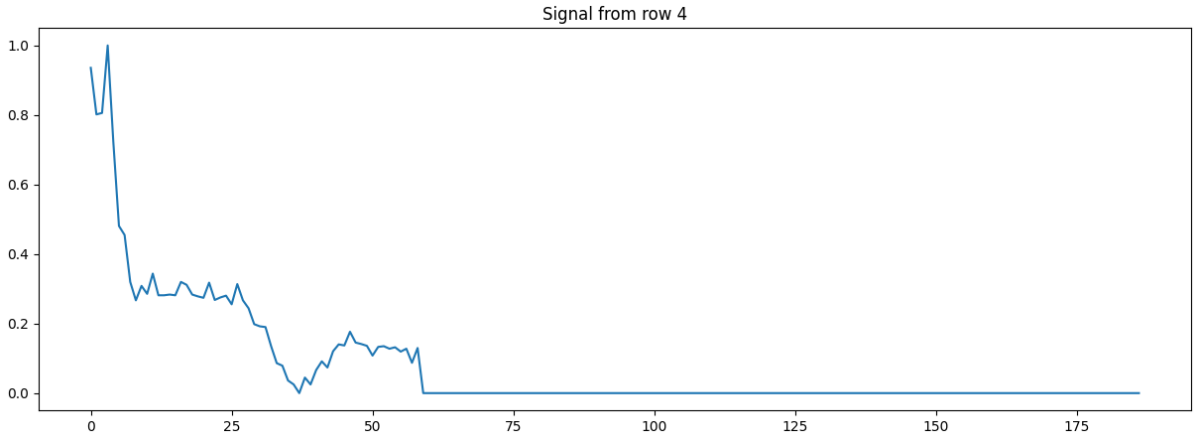

Figure 6. Abnormal ECG signal from PTBDB

The dataset is then divided in training, test and validation datasets.

#Split dataset in train, test and validation dataset

Step 1: Split the DataFrame into training+validation and testing datasets

train_val_df, test_df = train_test_split(full_ptb_shuffled_balanced, test_size=0.2, random_state=42)

Step 2: Split the training+validation dataset into training and validation datasets

train_df, val_df = train_test_split(train_val_df, test_size=0.25, random_state=42) # 0.25 * 0.8 = 0.2 of the original data

Display the shapes of the resulting DataFrames

print("Training DataFrame shape:", train_df.shape)

print("Validation DataFrame shape:", val_df.shape)

print("Testing DataFrame shape:", test_df.shape)

3.1.2 MITBIH dataset for multiclass classification

The MIT-BIH Arrhythmia Database [39], available on PhysioNet [15], is a comprehensive collection of 48 half-hour ECG recordings from 47 subjects. It includes both normal and abnormal heart rhythms, making it a vital resource for the development and evaluation of arrhythmia detection algorithms. Each recording is sampled at 360 Hz, with annotations provided for each beat and rhythm. The dataset is widely used in biomedical research for training and testing machine learning models aimed at detecting cardiac arrhythmias The dataset categorizes ECGs into five classes: Normal Sinus Rhythm, Supraventricular Ectopic Beat, Ventricular Ectopic Beat, Fusion Beat, and a category for unknown beats. These are labelled as follows: 'N' for Normal, 'S' for SEB, 'V' for VEB, 'F' for FB, and 'Q' for unknown beats, with corresponding numerical labels [0, 1, 2, 3, 4]. It is available on Kaggle [38] as two separate CSV files: one for the training dataset and another for the test dataset.

The first step is to import both csv files into panda data frames:

mitbih_train = pd.read_csv("/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/ECG Datasets/mitbih_train.csv")

mitbih_test = pd.read_csv("/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/ECG Datasets/mitbih_test.csv")

The last row (index 187) holds the classes 0, 1, 2, 3 and 4 for different ECG signals.

The initial number of classes is imbalanced in the training dataset, containing a large number of normal ECG samples than the other classes:

mitbih_train[187]=mitbih_train[mitbih_train.columns[187]].astype(int) # Use the correct column index from the DataFrame

balance=mitbih_train[mitbih_train.columns[187]].value_counts()

print(balance)

0.0 72470

4.0 6431

2.0 5788

1.0 2223

3.0 641

Name: count, dtype: int64

The class distribution is visualized using a bar chart:

```
# Sample data
# 0 - “N” for normal heartbeats.
# 1 - “S” for supra-ventricular premature.
# 2 - “V” for ventricular escape.
# 3 - “F” for fusion of ventricular and normal.
# 4 - “Q” for unclassified heartbeats
labels = ['n', 'q', 'v', 's', 'f']
colors = ['red', 'green', 'blue', 'skyblue', 'orange']
# Create a figure with a specific size
plt.figure(figsize=(20, 10))
# Create a bar chart
plt.bar(labels, balance, color=colors)
# Add titles and labels
plt.title('Bar Chart of Categories', fontsize=20)
plt.xlabel('Categories', fontsize=15)
plt.ylabel('Values', fontsize=15)
# Display the chart
plt.show()
```

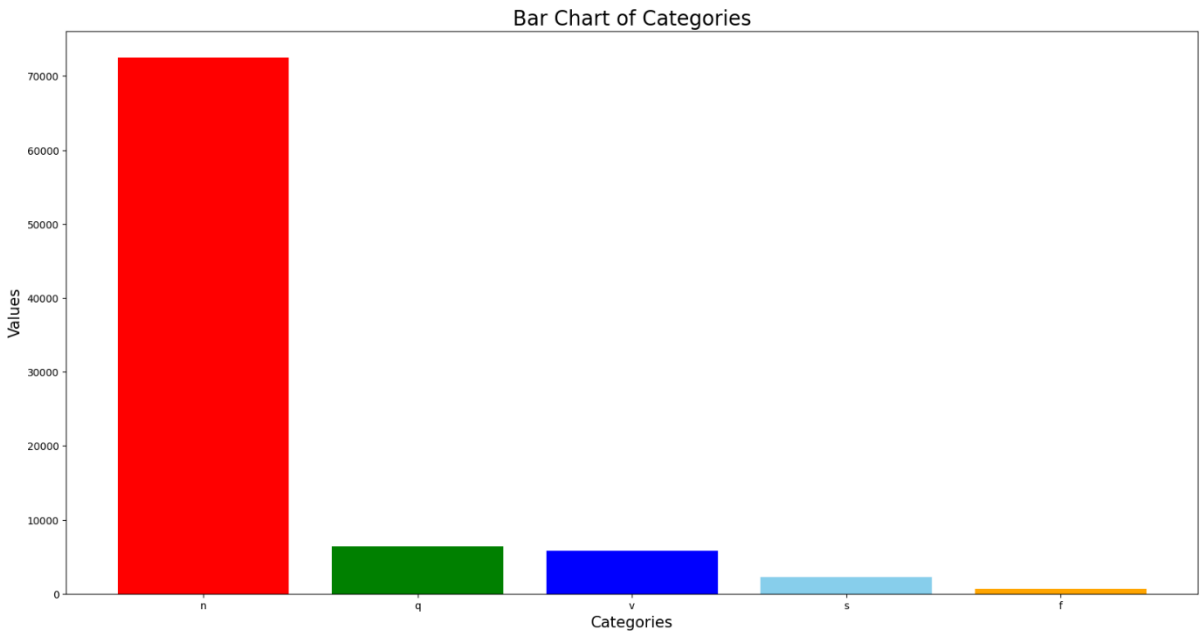

Figure 7. Initial balance of the MITBIH training dataset

Therefore, the data must be reorganized to balance the number of samples of each class:

```
from sklearn.utils import resample

df_1=mitbih_train[mitbih_train[187]==1]

df_2=mitbih_train[mitbih_train[187]==2]

df_3=mitbih_train[mitbih_train[187]==3]

df_4=mitbih_train[mitbih_train[187]==4]

df_0=(mitbih_train[mitbih_train[187]==0]).sample(n=20000,random_state=42)

df_1_upsample=resample(df_1,replace=True,n_samples=20000,random_state=123)

df_2_upsample=resample(df_2,replace=True,n_samples=20000,random_state=124)

df_3_upsample=resample(df_3,replace=True,n_samples=20000,random_state=125)

df_4_upsample=resample(df_4,replace=True,n_samples=20000,random_state=126)mitbih_
train_df=pd.concat([df_0,df_1_upsample,df_2_upsample,df_3_upsample,df_4_upsample])
```

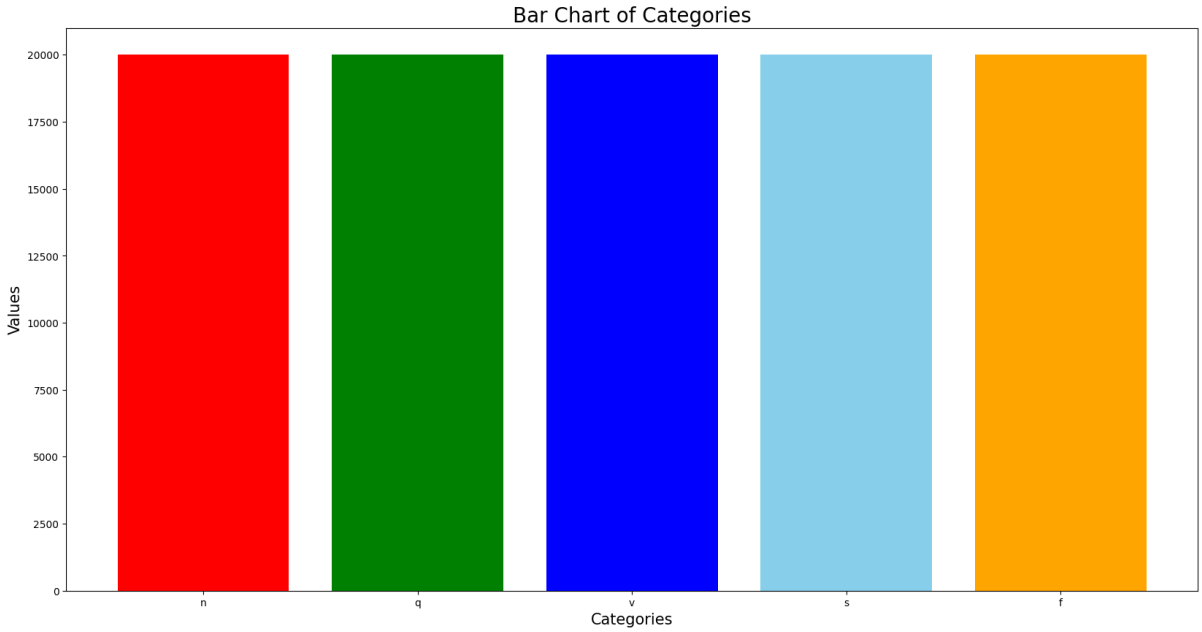

Figure 8. Balanced MITBIH training dataset bar chart.

Then, an ECG sample of each category is plotted using the **matplotlib**:

#Display table with some examples

examples=mitbih_train_df.groupby(187,group_keys=False).apply(lambda mitbih_train_df : mitbih_train_df.sample(1))

plt.plot(examples.iloc[0,:186])

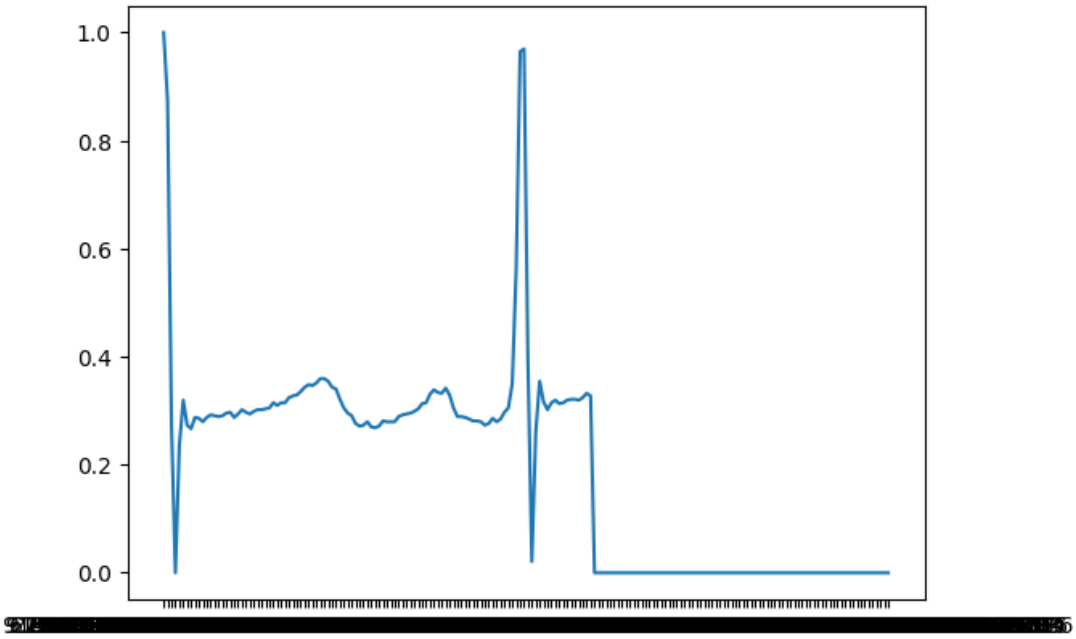

Figure 9. Normal ECG signal from MITBIH training dataset

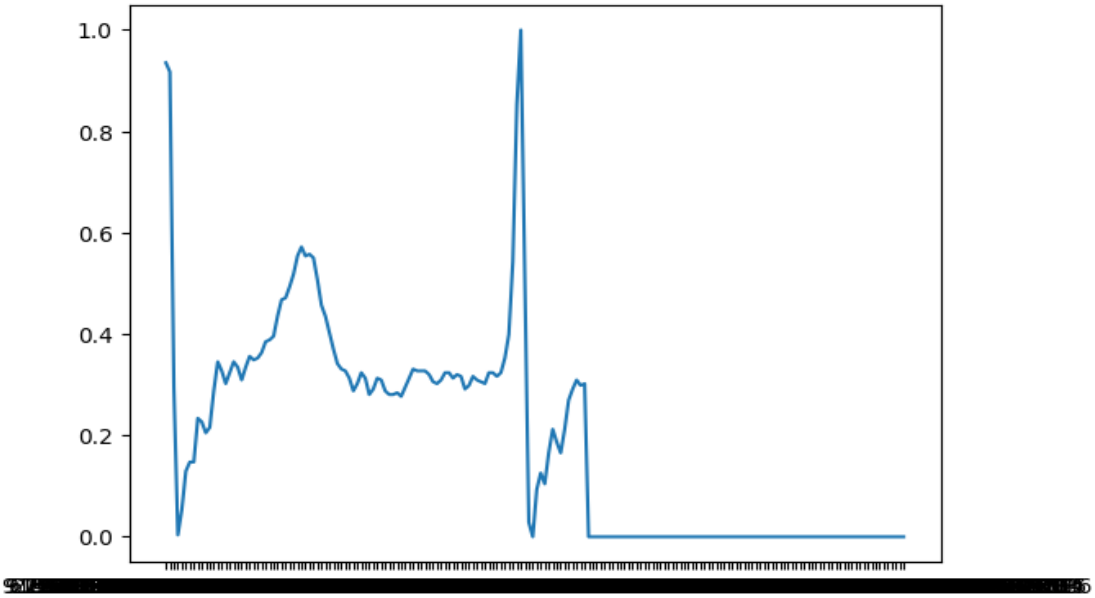

Figure 10. SEB ECG signal from MITBIH training dataset.

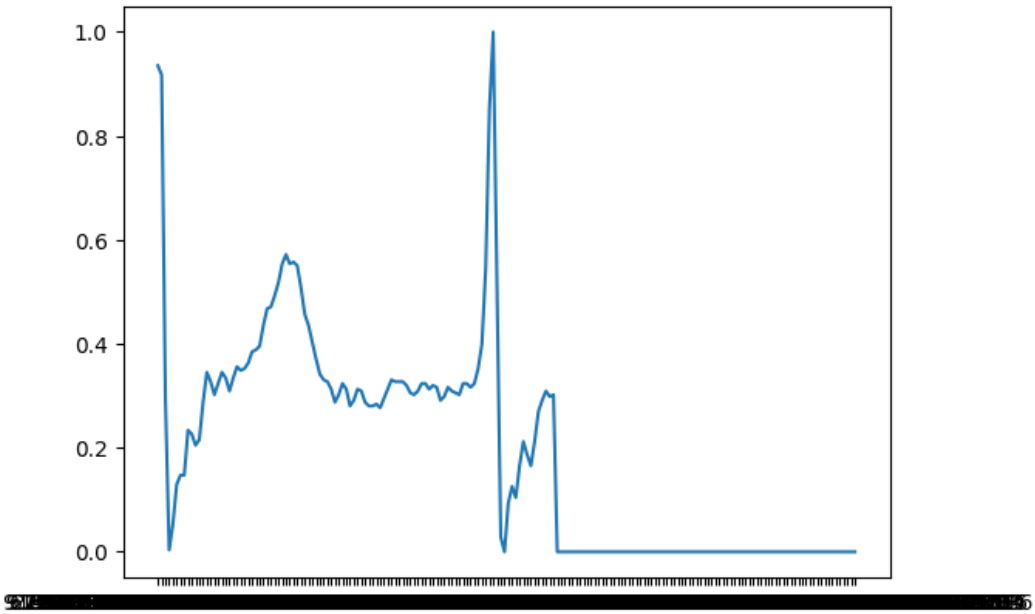

Figure 11. VEB ECG signal from MITBIH ECG dataset.

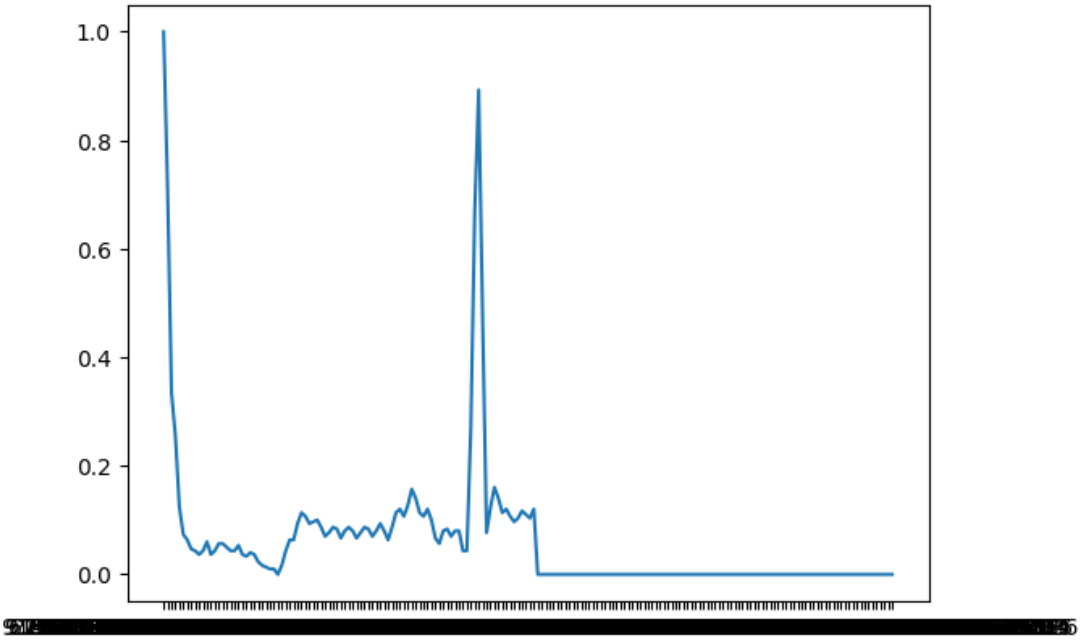

Figure 12. FB ECG signal from MITBIH training dataset.

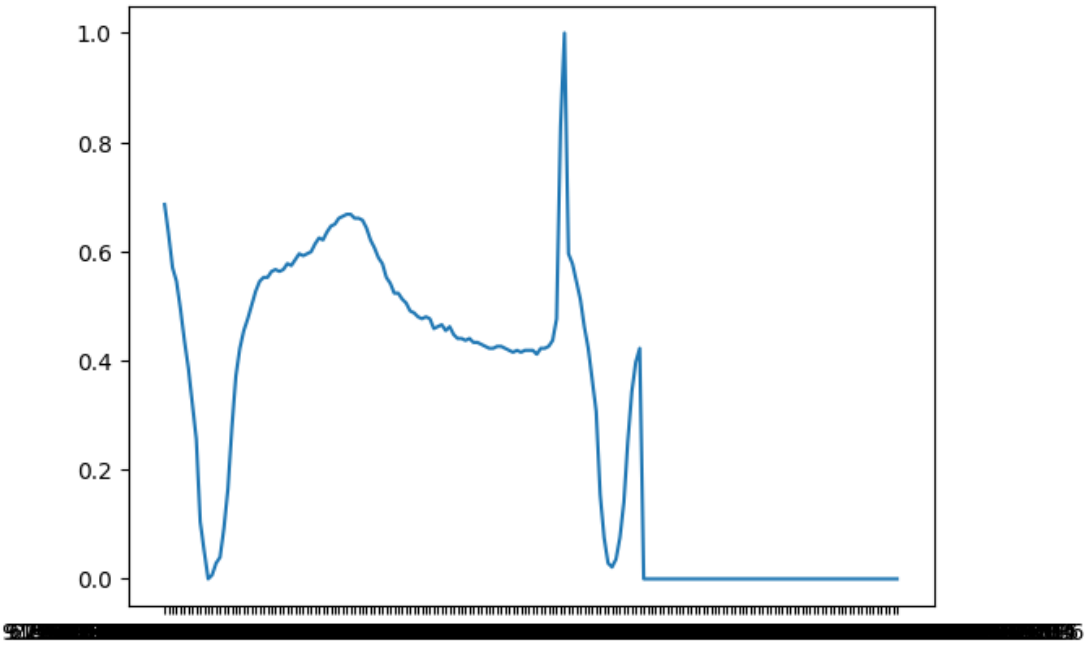

Figure 13. Unknown ECG signal from MITBIH training dataset.

Then, the values of X_train, X_test, y_train and y_test must be defined and reformatted to fit the NNs.

target_train = mitbih_train_df.iloc[:, 187] # Select column at index 187 mitbih_test_df = mitbih_test target_test = mitbih_test_df.iloc[:, 187] # Select column at index 187 y_train = to_categorical(target_train) y_test = to_categorical(target_test)

X_train=mitbih_train_df.iloc[:,:186].values X_test=mitbih_test_df.iloc[:,:186].values X_train = X_train.reshape(len(X_train), X_train.shape[1],1) X_test = X_test.reshape(len(X_test), X_test.shape[1],1)

The target labels (heart beat classes) are selected from the 187th column of both the training (mitbih_train_df) and test (mitbih_test_df) datasets. These labels are then converted to one-hot encoded format using to_categorical. The features (X_train and X_test) are extracted from all columns except the last (up to column 186) and reshaped to include a single channel to fit models' requirements.

4.2 Base Deep Learning models

Three base models have been developed, inspired by the most effective approaches identified in the literature review.

4.2.1 CNN model

The **CNN_model** defines a CNN model that creates and trains a Convolutional Neural Network (CNN) for a classification problem.

The model architecture is defined as follows:

- The input shape is determined by **X_train.shape[1]**, representing the number of features per sample.
- The model has two 1D convolutional layers (Convolution1D) with ReLU activation, each followed by batch normalization (BatchNormalization) and max-pooling layers (MaxPool1D).
- The output of the final convolutional layer is flattened (Flatten), then passed through two fully connected (dense) layers with 64 and 32 units, respectively.
- The final layer (Dense) contains either 2 units with a softmax activation function for binary classification or 5 units with a softmax activation function for multi-class classification, depending on the specific task.
- The model is compiled with the Adam optimizer, categorical cross-entropy loss, and accuracy as the evaluation metric.

- Early stopping (EarlyStopping) is used to stop training if the validation loss does not improve for 3 consecutive epochs.
- ModelCheckpoint saves the best model based on the validation loss.
- The model is trained for 5 epochs with a batch size of 32, using the training data (X_train, y_train) and validating on test data (X_test, y_test).

The final trained model is saved to a file, and the training history, which includes metrics like loss and accuracy over epochs, is stored as a JSON file. The function returns both the trained model and the training history, allowing for further analysis and use in subsequent tasks.

4.2.2 LSTM model

The **LSTM_model** function defines a model that creates and trains a Long Short-Term Memory (LSTM) neural network for ECG classification. The model architecture is defined as follows:

- The input shape is determined by the number of time steps (timesteps) and the number of features (input_dim) per sample.
- The model consists of two LSTM layers. The first LSTM layer has 64 units and returns sequences (meaning it passes the entire sequence to the next LSTM layer). The second LSTM layer also has 64 units and does not return sequences.
- Each LSTM layer is followed by batch normalization to stabilize and accelerate the training process.
- After the LSTM layers, there are two dense (fully connected) layers with 64 and 32 units, respectively, using ReLU activation.
- The final layer (Dense) contains either 2 units with a softmax activation function for binary classification or 5 units with a softmax activation function for multi-class classification, depending on the specific task.
- The model is compiled using the Adam optimizer, categorical cross-entropy loss (appropriate for multi-class classification), and accuracy as the evaluation metric.
- The callbacks define the Early Stopping stops training if the validation loss doesn't improve for 8 consecutive epochs, and the Model Checkpoint saves the best model during training based on the validation loss.
- The model is trained for 5 epochs with a batch size of 32, using the provided training and validation data.

The final trained model, including its architecture, weights, and optimizer state, is saved to a file, while the training history, capturing metrics like loss and accuracy over the epochs, is stored as a JSON file. The function then returns both the trained model and the training history, allowing for further analysis or use in subsequent tasks.

4.2.3 Transformer-based model

The **transformer_encoder** function defines a model that creates and trains a transformed-based neural network for ECG classification. The model is defined in three steps:

It implements Layer Normalization and Multi-Head Attention layers are applied to the inputs, followed by a residual connection and feed-forward layers using Conv1D. This is repeated for a specified number of transformer blocks.

The Model Architecture is defined as follows:

- The model begins with an input layer, passes through several transformer encoder blocks, and is followed by a global average pooling layer.
- Fully connected (Dense) layers with dropout are added before the final softmax layer for classification.
- The model is compiled with the Adam optimizer and categorical cross-entropy loss.
- Early stopping and model checkpoint callbacks are used during training to prevent overfitting and save the best model.

After training, the entire model and its training history are saved to files for future use. The **train_and_save_transformer_model** function is responsible for compiling, training, and saving the model along with its training history. This setup is designed to efficiently handle time-series data, leveraging the power of transformers for capturing complex patterns and dependencies.

3.2 Bagging technique implementation

To apply bagging technique, a bagging ensemble model is created by averaging the predictions from three different models: CNN, LSTM, and Transformer. It evaluates the performance of a bagging ensemble model by first converting the predicted probabilities (ensemble_multi_predictions) and the true labels (y_test) into class labels using **np.argmax**. It then calculates four key metrics: accuracy, precision, recall, and F1 score, using the **accuracy_score, precision_score, recall_score**, and **f1_score** functions from **sklearn.metrics**, with a weighted average for multi-class data. Finally, the metrics are printed out, providing a comprehensive assessment of the model's performance.

ensemble_binary_predictions = np.mean([cnn_pred, lstm_pred, transformer_pred], axis=0)

4. Design and Implementation

For this experiment, a variety of essential libraries have been used for data manipulation, visualization, machine learning, and deep learning.

- **NumPy and Pandas**: Handle data arrays and dataframes.
- **Matplotlib and Seaborn**: Create plots and visualizations.
- **Scikit-learn**: Provides tools for model selection, evaluation, and machine learning algorithms like RandomForestClassifier.
- **TensorFlow and Keras**: Frameworks for building and training deep learning models, with utilities for defining neural network layers, handling callbacks, and saving/loading models.
- **Warnings**: Suppress warnings for a cleaner output.

This code was executed in Google Colab using a T4 GPU runtime, which accelerates the training of deep learning models.

5. Results and Discussion

The results obtained were based on a limited number of epochs for each model due to the technical constraints of Google Colab. While Colab offers substantial computational resources, training complex models like those required for the MITBIH dataset can be time-consuming due to the complexity and size of the data, leading to extended training times.

5.1 Results of binary classification on PTB dataset

The binary classification experiment conducted using the PTB dataset yielded the following results across ten epochs for each model:

The CNN model performed exceptionally well in the binary classification task, achieving an accuracy of 95.18%. The precision and recall are closely aligned, indicating that the model has a balanced performance in terms of predicting both classes accurately. The F1 score of 0.9518 confirms this balance, showing that the model handles false positives and false negatives effectively. The model shows significant improvement after the third epoch, where accuracy jumps from around 68.99% to 93.47%, with corresponding improvements in loss. This suggests that the model required a few epochs to start converging effectively. The validation accuracy stabilizes around 95% by the 10th epoch, indicating that the model generalizes well to unseen data.

Table 1. CNN for binary classification over 10 epochs.

Epoch	**accuracy**	**loss**	**val_accuracy**	**val_loss**
1	0.624	0.649	0.4935	0.7074
2	0.6852	0.5983	0.512	0.7175
3	0.6899	0.593	0.6745	0.6509
4	0.9347	0.1759	0.9179	0.2199
5	0.9381	0.1602	0.9444	0.1878
6	0.9506	0.1363	0.9463	0.1679
7	0.9486	0.1339	0.9395	0.1738
8	0.9558	0.1168	0.9296	0.2054
9	0.9589	0.1142	0.9629	0.133
10	0.9608	0.1044	0.9518	0.1515

The LSTM model did not perform as well as the CNN model in binary classification. With an accuracy of 67.08%, the model's performance is modest. The higher precision (74.20%) compared to recall (67.08%) suggests that while the model is good at identifying true positives, it misses a significant number of actual positives, leading to a lower recall. The F1 score reflects this imbalance. The model shows steady but slow improvement over the epochs. However, it struggles to reach higher accuracy, plateauing around 70%. This indicates that the LSTM might be less effective at capturing the temporal dependencies in this binary classification task compared to CNN.

Table 2. LSTM for binary classification over 10 epochs.

Epoch	**accuracy**	**loss**	**val_accuracy**	**val_loss**
1	0.624	0.649	0.4935	0.7074
2	0.6852	0.5983	0.512	0.7175
3	0.6899	0.593	0.6745	0.6509
4	0.6991	0.5831	0.5726	0.6537
5	0.7003	0.5704	0.6708	0.5922
6	0.6991	0.5831	0.5726	0.6537
7	0.7003	0.5704	0.6708	0.5922
8	0.6998	0.5666	0.622	0.6278
9	0.7052	0.5698	0.6566	0.6029
10	0.7054	0.5604	0.4978	0.8174

The Transformer model performed the worst among the three models in binary classification, with an accuracy of 57.01%. The precision and recall values are both low, indicating that the model struggles with both classes, leading to a poor F1 score of 0.5497. This suggests that the Transformer model might not be well-suited for this particular task or might require further tuning or more data to perform effectively. The model shows minimal improvement over epochs, with accuracy lingering around 50-60%. The validation accuracy remains low, indicating poor generalization.

Table 3. Transformer-based model for binary classification over 10 epochs.

Epoch	loss	accuracy	val_loss	val_accuracy
1	0.6922	0.5163	0.6898	0.512
2	0.6862	0.5548	0.6874	0.5318
3	0.6813	0.5591	0.6844	0.5479
4	0.6793	0.5664	0.6814	0.5707
5	0.6817	0.5592	0.6866	0.5269
6	0.678	0.5694	0.6811	0.5522
7	0.6758	0.5775	0.6794	0.5868
8	0.6759	0.5781	0.6784	0.6022
9	0.679	0.5738	0.6787	0.6109
10	0.6748	0.5833	0.6794	0.5701

The Bagging ensemble model, which combines the predictions of the three models, performed very well with an accuracy of 92.90%. The ensemble approach seems to mitigate the weaknesses of the individual models, leading to a balanced and high performance across all metrics. The slight drop in performance compared to the CNN could be due to the weaker performance of the LSTM and Transformer models diluting the ensemble's overall effectiveness.

Table 4. Metrics for binary ECG classification on PTB dataset

Model	Accuracy	Precision	Recall	F1 Score
CNN	*0.9518*	*0.9533*	*0.9518*	*0.9518*
LSTM	*0.6708*	*0.7420*	*0.6708*	*0.6466*
Transformer	*0.5701*	*0.5897*	*0.5701*	*0.5497*
Bagging Ensemble	*0.9290*	*0.9356*	*0.9290*	*0.9288*

5.2 Results of multiclass classification on MITBIH dataset

The multiclass classification experiment conducted with the MITBIH dataset produced the following results over ten epochs for each model:

For multiclass classification, the CNN model continues to perform exceptionally well, with an accuracy of 96.74%. The high precision, recall, and F1 scores indicate that the model effectively discriminates between multiple classes, handling the complexity of the task with ease. The model shows strong performance from the first epoch, with accuracy improving steadily and validation accuracy remaining high, confirming good generalization. The final validation accuracy of 96.74% is consistent with the test accuracy, showing that the model is robust and well-trained.

Table 5. CNN for multiclass classification over 10 epochs.

Epoch	**accuracy**	**loss**	**val_accuracy**	**val_loss**
1	0.8624	0.3825	0.8638	0.3923
2	0.9349	0.1878	0.9297	0.2122
3	0.956	0.1308	0.9225	0.2383
4	0.9879	0.0372	0.9573	0.1511
5	0.9906	0.03	0.9695	0.1346
6	0.9918	0.0256	0.9662	0.1439
7	0.9929	0.0227	0.9709	0.1347
8	0.9944	0.0185	0.9635	0.1554
9	0.9942	0.0187	0.9741	0.1327
10	0.9955	0.0156	0.9674	0.1848

In the multiclass setting, the LSTM model performs better than in binary classification but still lags behind the CNN. An accuracy of 79.20% is respectable, and the high precision suggests that when the model makes a positive prediction, it is often correct. However, the recall is lower, which means it fails to identify a significant number of true instances across multiple classes. The F1 score of 0.8472 shows a decent but not outstanding balance. The model's performance improves over the epochs, but it seems to plateau earlier than the CNN. This suggests that while LSTM can capture some of the temporal dependencies in multiclass tasks, it might not be as effective as CNN in handling the spatial features inherent in ECG data.

Table 6. LSTM for multiclass classification over 10 epochs.

Epoch	**accuracy**	**loss**	**val_accuracy**	**val_loss**
1	0.6471	0.9097	0.7684	0.6556
2	0.8467	0.4334	0.7968	0.5676
3	0.8766	0.3498	0.792	0.5604
4	0.9918	0.0256	0.9662	0.1439
5	0.9929	0.0227	0.9709	0.1347
6	0.9944	0.0185	0.9635	0.1554
7	0.9942	0.0187	0.9741	0.1327
8	0.9955	0.0156	0.9674	0.1848
9	0.9918	0.0256	0.9662	0.1439
10	0.9929	0.0227	0.9709	0.1347

The Transformer model performs poorly in the multiclass classification task, with an accuracy of only 16.87%. The precision is relatively high at 77.69%, but the recall is very low, indicating that the model is failing to identify true instances across most classes. The low F1 score reflects the poor overall performance. This result suggests that the Transformer model is not effective in handling this type of ECG data for multiclass classification. The model's performance shows minimal improvement over epochs, with validation accuracy remaining low. The loss values suggest that the model is not learning effectively from the data, possibly due to inadequate feature extraction or model complexity issues.

Table 7. Transformer-based model for multiclass classification over 10 epochs.

Epoch	loss	accuracy	val_loss	val_accuracy
1	1.4313	0.3676	1.4641	0.1641
2	1.3773	0.3918	1.4508	0.2077
3	1.3604	0.3998	1.4745	0.1562
4	1.3458	0.4075	1.497	0.1321
5	1.3369	0.4142	1.5023	0.1298
6	1.3313	0.4175	1.4503	0.1142
7	1.3282	0.4217	1.4487	0.1456
8	1.3264	0.4226	1.4693	0.1615
9	1.3246	0.4254	1.4532	0.1648
10	1.3237	0.4245	1.4337	0.1687

The Bagging ensemble model also performs very well in the multiclass classification task, with an accuracy of 95.78%. This model balances the strengths of the individual models, resulting in high precision, recall, and F1 scores. While it slightly underperforms compared to the CNN, it still provides robust results, likely benefiting from the diversity of the models in the ensemble.

Table 8. Metrics for multiclass ECG classification on MITBIH dataset

Model	*Accuracy*	*Precision*	*Recall*	*F1 Score*
CNN	*0.9674*	*0.9717*	*0.9674*	*0.9689*
LSTM	*0.7920*	*0.9321*	*0.7920*	*0.8472*
Transformer	*0.1687*	*0.7769*	*0.1687*	*0.2184*
Bagging Ensemble	*0.9578*	*0.9683*	*0.9578*	*0.9616*

6. Conclusions and Recommendations

Overall, the CNN model consistently outperformed the LSTM and Transformer models in both binary and multiclass ECG classification tasks, demonstrating a strong capability in handling the complexities of ECG data. For binary classification, the CNN achieved an impressive accuracy of 95.18%, with a precision of 95.33%, recall of 95.18%, and an F1 score of 95.18%. These metrics indicate that the CNN model effectively balances precision and recall, making it highly reliable for binary ECG classification.

In contrast, the LSTM model, while showing reasonable performance, struggled with recall, particularly in the multiclass classification task. Specifically, in binary classification, the LSTM had an accuracy of 67.08%, precision of 74.20%, recall of 67.08%, and an F1 score of 64.66%. This discrepancy between precision and recall suggests that the LSTM model may have difficulties capturing the full range of features necessary for accurate classification, leading to a higher number of false negatives.

The Transformer model performed poorly across both tasks, with an accuracy of 57.01% in binary classification and a notably low 16.87% in multiclass classification. Its low F1 scores of 54.97% for binary and 21.84% for multiclass tasks indicate significant challenges in both precision and recall. These results suggest that the Transformer model, in its current configuration, is not well-suited for this type of ECG data and may require substantial tuning or redesign to improve its performance.

The Bagging ensemble model, which combines the predictions of the CNN, LSTM, and Transformer models, demonstrated strong performance, particularly in the multiclass classification task. For binary classification, the Bagging model achieved an accuracy of 92.90%, with precision and recall both at 93.56% and 92.90%, respectively. In multiclass classification, it recorded an accuracy of 95.78%, precision of 96.83%, recall of 95.78%, and an F1 score of 96.16%. These metrics highlight the ensemble model's ability to mitigate the weaknesses of individual models, leading to more robust and reliable performance.

In conclusion, the CNN model stands out as the most effective model for ECG classification tasks, particularly when dealing with complex and high-dimensional data. The Bagging ensemble approach further enhances performance by leveraging the strengths of multiple models, making it a viable strategy for improving classification accuracy and robustness. However, the LSTM and Transformer models may require further tuning or alternative configurations to achieve better results, particularly in capturing the intricate features of ECG data.

To enhance the performance and applicability of the ECG classification models, further work should focus on optimizing and fine-tuning the LSTM and Transformer models, exploring advanced CNN architectures, and applying sophisticated data augmentation and preprocessing techniques. Additionally, experimenting with diverse ensemble strategies, such as Boosting or Stacking, and incorporating other models like SVMs or Random Forests could

improve overall accuracy. Leveraging transfer learning, enhancing model interpretability through Explainable AI techniques, and conducting a detailed error analysis are also critical steps. Moreover, validating models across different datasets, optimizing for real-time deployment, and integrating clinical metadata will ensure robustness and practical utility in clinical settings. Lastly, incorporating longitudinal analysis for temporal consistency could further improve predictive capabilities.

7. Reflection

As an engineer working in a defibrillator company, I am particularly interested in the application of computational methods to the classification and recognition of electrocardiograms. When it came time to choose a topic for this dissertation, I saw this as a highly productive research area with direct and practical applications.

Once the research topic was finalized, the subsequent steps in the process gradually took shape. Each day brought new learning opportunities, whether it involved discovering relevant data in the field or experimenting with code in Colab or Jupyter to test new implementations. This iterative process not only deepened my understanding of deep learning models but also highlighted the complexities involved in adapting these models to medical data.

Despite the abundance of articles on DL model research for ECG classification tasks, one of the biggest challenges remains the limited availability of datasets, primarily due to the sensitive nature of ECG data. Accessing comprehensive and diverse datasets is crucial for training robust models, yet the ethical and privacy concerns surrounding medical data often limit what is available. This constraint pushed me to be more resourceful, focusing on maximizing the potential of the datasets I had access to and ensuring that the models were trained and validated as effectively as possible.

However, it has been rewarding to explore various DL models, observe how different designs respond to experiments, and witness the improvement in results as more data was gathered and the models were refined. The process of iterating on these models, tweaking parameters, and integrating feedback from each experiment has been both challenging and fulfilling. Each small breakthrough, whether it was a slight increase in accuracy or a more efficient model architecture, reinforced my belief in the potential of machine learning to significantly impact ECG classification.

This dissertation has not only enhanced my technical skills but also broadened my perspective on the intersection of engineering and healthcare. The experience has solidified my passion for this field and my commitment to leveraging computational techniques to advance cardiac health monitoring and diagnostics. Looking back, I am proud of the progress made and excited about the possibilities for future research and applications in this vital area.

8. References

[1] ‘Cardiovascular diseases (CVDs)’. Accessed: Jul. 08, 2024. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] P. A. McCullough, ‘Coronary Artery Disease’, *Clin. J. Am. Soc. Nephrol.*, vol. 2, no. 3, p. 611, May 2007, doi: 10.2215/CJN.03871106.

[3] O. A. Carretero and S. Oparil, ‘Essential Hypertension’, *Circulation*, vol. 101, no. 3, pp. 329–335, Jan. 2000, doi: 10.1161/01.CIR.101.3.329.

[4] A. Groenewegen, F. H. Rutten, A. Mosterd, and A. W. Hoes, ‘Epidemiology of heart failure’, *Eur. J. Heart Fail.*, vol. 22, no. 8, pp. 1342–1356, 2020, doi: 10.1002/ejhf.1858.

[5] S. M. Anwar, M. Gul, M. Majid, and M. Alnowami, ‘Arrhythmia Classification of ECG Signals Using Hybrid Features’, *Comput. Math. Methods Med.*, vol. 2018, no. 1, p. 1380348, 2018, doi: 10.1155/2018/1380348.

[6] ‘The evolution of pacemakers | IEEE Journals & Magazine | IEEE Xplore’. Accessed: Jul. 11, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1636350

[7] W. B. Fye, ‘A history of the origin, evolution, and impact of electrocardiography’, *Am. J. Cardiol.*, vol. 73, no. 13, pp. 937–949, May 1994, doi: 10.1016/0002-9149(94)90135-x.

[8] H. Narotamo, M. Dias, R. Santos, A. V. Carreiro, H. Gamboa, and M. Silveira, ‘Deep learning for ECG classification: A comparative study of 1D and 2D representations and multimodal fusion approaches’, *Biomed. Signal Process. Control*, vol. 93, p. 106141, Jul. 2024, doi: 10.1016/j.bspc.2024.106141.

[9] X. Liu, H. Wang, Z. Li, and L. Qin, ‘Deep learning in ECG diagnosis: A review’, *Knowl.-Based Syst.*, vol. 227, p. 107187, Sep. 2021, doi: 10.1016/j.knosys.2021.107187.

[10] I. Tomašić and R. Trobec, ‘Electrocardiographic Systems With Reduced Numbers of Leads—Synthesis of the 12-Lead ECG’, *IEEE Rev. Biomed. Eng.*, vol. 7, pp. 126–142, 2014, doi: 10.1109/RBME.2013.2264282.

[11] D. D. Finlay, C. D. Nugent, J. G. Kellett, M. P. Donnelly, P. J. McCullagh, and N. D. Black, ‘Synthesising the 12-lead electrocardiogram: Trends and challenges’, *Eur. J. Intern. Med.*, vol. 18, no. 8, pp. 566–570, Dec. 2007, doi: 10.1016/j.ejim.2007.04.011.

[12] ‘QRS complex detection and arrhythmia classification using SVM | IEEE Conference Publication | IEEE Xplore’. Accessed: Jul. 24, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/7437915

[13] ‘Deep learning and the electrocardiogram: review of the current state-of-the-art | EP Europace | Oxford Academic’. Accessed: Jul. 24, 2024. [Online]. Available: https://academic.oup.com/europace/article/23/8/1179/6132071
[14] ‘Application of deep learning techniques for heartbeats detection using ECG signals-analysis and review - ScienceDirect’. Accessed: Jul. 24, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482520301104
[15] G. B. Moody and R. G. Mark, ‘MIT-BIH Arrhythmia Database’. physionet.org, 1992. doi: 10.13026/C2F305.
[16] P. Wagner *et al.*, ‘PTB-XL, a large publicly available electrocardiography dataset’, *Sci. Data*, vol. 7, no. 1, p. 154, May 2020, doi: 10.1038/s41597-020-0495-6.
[17] V. Tihonenko, A. Khaustov, S. Ivanov, and A. Rivin, ‘St.-Petersburg Institute of Cardiological Technics 12-lead Arrhythmia Database’. physionet.org, 2007. doi: 10.13026/C2V88N.
[18] N. Iyengar, C.-K. Peng, R. Morin, A. L. Goldberger, and L. A. Lipsitz, ‘Fantasia Database’. physionet.org, 1994. doi: 10.13026/C2RG61.
[19] Q. Xiao *et al.*, ‘Deep Learning-Based ECG Arrhythmia Classification: A Systematic Review’, *Appl. Sci.*, vol. 13, no. 8, Art. no. 8, Jan. 2023, doi: 10.3390/app13084964.
[20] E. A. Ashley, E. Ashley, and J. Niebauer, *Cardiology Explained*. Remedica, 2004.
[21] ‘Ectopic Supraventricular Arrhythmias - Cardiovascular Disorders’, MSD Manual Professional Edition. Accessed: Aug. 26, 2024. [Online]. Available: https://www.msdmanuals.com/en-gb/professional/cardiovascular-disorders/specific-cardiac-arrhythmias/ectopic-supraventricular-arrhythmias
[22] A. J. Moss, H. T. Davis, J. DeCamilla, and L. W. Bayer, ‘Ventricular ectopic beats and their relation to sudden and nonsudden cardiac death after myocardial infarction.’, *Circulation*, vol. 60, no. 5, pp. 998–1003, Nov. 1979, doi: 10.1161/01.CIR.60.5.998.
[23] ‘Ventricular Fusion Beats’. Accessed: Aug. 26, 2024. [Online]. Available: https://www.ahajournals.org/doi/epdf/10.1161/01.CIR.26.5.880
[24] F. Mi, B. Li, X. Cheng, Y. Zhao, M. Li, and J. Jing, ‘Classification and Processing of MIT-BIH Arrhythmia-Based on BP Algorithm’, in *2023 International Conference on Intelligent Supercomputing and BioPharma (ISBP)*, Jan. 2023, pp. 72–76. doi: 10.1109/ISBP57705.2023.10061303.
[25] G. Ekİnci, E. Kardeş, H. Güvenkaya, and P. Karagöz, ‘Investigating the preprocessing methods in ECG analysis’, in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, Jun. 2021, pp. 1–4. doi: 10.1109/SIU53274.2021.9477702.
[26] S. Śmigiel, K. Pałczyński, and D. Ledziński, ‘ECG Signal Classification Using Deep Learning Techniques Based on the PTB-XL Dataset’, *Entropy*, vol. 23, no. 9, Art. no. 9, Sep. 2021, doi: 10.3390/e23091121.
[27] Z. Chen *et al.*, ‘A novel imbalanced dataset mitigation method and ECG classification model based on combined 1D_CBAM-autoencoder and lightweight CNN model’, *Biomed. Signal Process. Control*, vol. 87, p. 105437, Jan. 2024, doi: 10.1016/j.bspc.2023.105437.
[28] ‘Applied Sciences | Free Full-Text | A Shallow Domain Knowledge Injection (SDK-Injection) Method for Improving CNN-Based ECG Pattern Classification’. Accessed: Aug. 06, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1307
[29] L. D. Sharma, J. Rahul, A. Aggarwal, and V. K. Bohat, ‘An improved cardiac arrhythmia classification using stationary wavelet transform decomposed short duration QRS segment and Bi-LSTM network’, *Multidimens. Syst. Signal Process.*, vol. 34, no. 2, pp. 503–520, Jun. 2023, doi: 10.1007/s11045-023-00875-x.

[30] M. Karri and C. S. R. Annavarapu, 'A real-time embedded system to detect QRS-complex and arrhythmia classification using LSTM through hybridized features', *Expert Syst. Appl.*, vol. 214, p. 119221, Mar. 2023, doi: 10.1016/j.eswa.2022.119221.
[31] S. Boda, M. Mahadevappa, and P. Kumar Dutta, 'An automated patient-specific ECG beat classification using LSTM-based recurrent neural networks', *Biomed. Signal Process. Control*, vol. 84, p. 104756, Jul. 2023, doi: 10.1016/j.bspc.2023.104756.
[32] H. El-Ghaish and E. Eldele, 'ECGTransForm: Empowering adaptive ECG arrhythmia classification framework with bidirectional transformer', *Biomed. Signal Process. Control*, vol. 89, p. 105714, Mar. 2024, doi: 10.1016/j.bspc.2023.105714.
[33] Y. Dong, M. Zhang, L. Qiu, L. Wang, and Y. Yu, 'An Arrhythmia Classification Model Based on Vision Transformer with Deformable Attention', *Micromachines*, vol. 14, no. 6, Art. no. 6, Jun. 2023, doi: 10.3390/mi14061155.
[34] A. Varghese, S. Kamal, and J. Kurian, 'Transformer-based temporal sequence learners for arrhythmia classification', *Med. Biol. Eng. Comput.*, vol. 61, no. 8, pp. 1993–2000, Aug. 2023, doi: 10.1007/s11517-023-02858-3.
[35] M. Maleki, 'Identification of cardiovascular diseases through ECG classification using wavelet transformation', Apr. 14, 2024, *arXiv*: arXiv:2404.09393. doi: 10.48550/arXiv.2404.09393.
[36] '(PDF) ECG Classification Exercise Health Analysis Algorithm Based on GRU and Convolutional Neural Network'. Accessed: Jul. 25, 2024. [Online]. Available: https://www.researchgate.net/publication/380088734_ECG_classification_exercise_health_analysis_algorithm_based_on_GRU_and_convolutional_neural_network
[37] P. Pławiak and U. R. Acharya, 'RETRACTED ARTICLE: Novel deep genetic ensemble of classifiers for arrhythmia detection using ECG signals', *Neural Comput. Appl.*, vol. 32, no. 15, pp. 11137–11161, Aug. 2020, doi: 10.1007/s00521-018-03980-2.
[38] 'ECG Heartbeat Categorization Dataset'. Accessed: Aug. 27, 2024. [Online]. Available: https://www.kaggle.com/datasets/shayanfazeli/heartbeat
[39] G. B. Moody and R. G. Mark, 'The impact of the MIT-BIH arrhythmia database', *IEEE Eng. Med. Biol. Mag. Q. Mag. Eng. Med. Biol. Soc.*, vol. 20, no. 3, pp. 45–50, 2001, doi: 10.1109/51.932724.

APPENDIX I Code: Deep Learning models

```python
#Create and store CNN model and history metrics
def CNN_model(X_train, y_train, X_test, y_test):
    im_shape = (X_train.shape[1], 1)
    inputs_cnn = Input(shape=(im_shape), name='inputs_cnn')

    # First Convolutional Layer
    conv1 = Convolution1D(32, (6), activation='relu')(inputs_cnn)
    pool1 = MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv1)

    # Second Convolutional Layer
    conv2 = Convolution1D(32, (3), activation='relu')(pool1)
    pool2 = MaxPool1D(pool_size=(2), strides=(2), padding="same")(conv2)

    # Flatten and Dense Layers
    flatten = Flatten()(pool2)
    dense1 = Dense(32, activation='relu')(flatten)
    main_output = Dense(2, activation='softmax', name='main_output')(dense1)  # Output layer for binary classification

    # Compile the model
    model = Model(inputs=inputs_cnn, outputs=main_output)
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    # Callbacks for early stopping and model checkpointing
    callbacks = [
        EarlyStopping(monitor='val_loss', patience=3),
        ModelCheckpoint(filepath='/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/CNN/best_model_CNN_binary.keras', monitor='val_loss', save_best_only=True)
    ]

    # Train the model
    history = model.fit(X_train, y_train, epochs=3, callbacks=callbacks, batch_size=32, validation_data=(X_test, y_test))

    # Save the final model
    model.save('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/CNN/final_cnn_model_binary.keras')

    # Save the training history to a JSON file
    with open('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/CNN/cnn_binary_training_history.json', 'w') as f:
        json.dump(history.history, f)

    return model, history
```

```python
[13] #Create and store LSTM model and history metrics
def LSTM_model(X_train, y_train, X_test, y_test):
    timesteps = X_train.shape[1]
    input_dim = X_train.shape[2]

    inputs_lstm = Input(shape=(timesteps, input_dim), name='inputs_lstm')
    lstm1 = LSTM(64, return_sequences=True)(inputs_lstm)
    lstm1 = BatchNormalization()(lstm1)
    lstm2 = LSTM(64)(lstm1)
    lstm2 = BatchNormalization()(lstm2)
    dense_end1 = Dense(64, activation='relu')(lstm2)
    dense_end2 = Dense(32, activation='relu')(dense_end1)
    main_output = Dense(2, activation='softmax', name='main_output')(dense_end2)

    model = Model(inputs=inputs_lstm, outputs=main_output)
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    callbacks = [
        EarlyStopping(monitor='val_loss', patience=8),
        ModelCheckpoint(filepath='/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/LSTM/best_model_LSTM_binary.keras', monitor='val_loss', save_best_only=True)
    ]

    history = model.fit(X_train, y_train, epochs=3, callbacks=callbacks, batch_size=32, validation_data=(X_test, y_test))

    # Save the entire model to a file
    model.save('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/LSTM/final_lstm_model_binary.keras')  # Save the model architecture, weights, and optimizer state

    # Save the training history to a JSON file
    with open('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/LSTM/lstm_binary_training_history.json', 'w') as f:
        json.dump(history.history, f)

    return model, history
```

```python
[14] #Transformer model

def transformer_encoder(inputs, head_size, num_heads, ff_dim, dropout=0):
    # Normalization and Attention
    x = layers.LayerNormalization(epsilon=1e-6)(inputs)
    x = layers.MultiHeadAttention(key_dim=head_size, num_heads=num_heads, dropout=dropout)(x, x)
    x = layers.Dropout(dropout)(x)
    res = x + inputs

    # Feed Forward Part
    x = layers.LayerNormalization(epsilon=1e-6)(res)
    x = layers.Conv1D(filters=ff_dim, kernel_size=1, activation="relu")(x)
    x = layers.Dropout(dropout)(x)
    x = layers.Conv1D(filters=inputs.shape[-1], kernel_size=1)(x)
    return x + res

def build_transformer_model(input_shape, num_classes, head_size=256, num_heads=4, ff_dim=4, num_transformer_blocks=4, mlp_units=[128], dropout=0.1):
    inputs = layers.Input(shape=input_shape)
    x = inputs
    for _ in range(num_transformer_blocks):
        x = transformer_encoder(x, head_size, num_heads, ff_dim, dropout)

    x = layers.GlobalAveragePooling1D()(x)  # Removed data_format="channels_first"
    for dim in mlp_units:
        x = layers.Dense(dim, activation="relu")(x)
        x = layers.Dropout(dropout)(x)
    outputs = layers.Dense(num_classes, activation="softmax")(x)
    return tf.keras.Model(inputs, outputs)

# Example to compile, train, save the model and history
def train_and_save_transformer_model(X_train, y_train, X_test, y_test, input_shape, num_classes):
    model = build_transformer_model(input_shape, num_classes)
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    callbacks = [
        tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=8),
        tf.keras.callbacks.ModelCheckpoint(filepath='/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/Transformer/best_transformer_model_binary.keras', monitor='val_loss', save_best_only=True)
    ]

    history = model.fit(X_train, y_train, epochs=1, callbacks=callbacks, batch_size=32, validation_data=(X_test, y_test))


    # Save the entire model to a file
    model.save('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/Transformer/final_transformer_model_binary.keras')  # Save the model architecture, weights, and optimizer state

    # Save the training history to a JSON file
    with open('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/Transformer/transformer_binary_training_history.json', 'w') as f:
        json.dump(history.history, f)

    return model, history
```

APPENDIX II Code: Bagging technique implementation

```
[ ] # Create the bagging ensemble by averaging the predictions

    # Load the models from the files
    cnn_model = load_model('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/CNN/final_cnn_model.keras')

    lstm_model = load_model('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/LSTM/final_lstm_model.keras')

    transformer_model = build_transformer_model(input_shape, num_classes)
    model.load_weights('/content/drive/MyDrive/UWTSD_MSc_SE_and_AI/Dissertation/Code/Transformer/final_transformer_model.keras (Unzipped Files)/model.weights.h5')
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    cnn_pred = cnn_model.predict(X_test)
    lstm_pred = lstm_model.predict(X_test)
    transformer_pred = transformer_model.predict(X_test)

    ensemble_binary_predictions = np.mean([cnn_pred, lstm_pred, transformer_pred], axis=0)

    685/685 ━━━━━━━━━━━━━━━━━━━━ 10s 14ms/step
    685/685 ━━━━━━━━━━━━━━━━━━━━ 48s 69ms/step
    685/685 ━━━━━━━━━━━━━━━━━━━━ 658s 959ms/step

[ ] # Convert predictions to class labels
    y_pred_classes = np.argmax(ensemble_multi_predictions, axis=1)
    y_test_classes = np.argmax(y_test, axis=1)

    # Evaluate the ensemble model
    from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

    accuracy = accuracy_score(y_test_classes, y_pred_classes)
    precision = precision_score(y_test_classes, y_pred_classes, average='weighted')
    recall = recall_score(y_test_classes, y_pred_classes, average='weighted')
    f1 = f1_score(y_test_classes, y_pred_classes, average='weighted')

    print(f'Bagging Ensemble Model metrics:\nAccuracy: {accuracy:.4f}\nPrecision: {precision:.4f}\nRecall: {recall:.4f}\nF1 Score: {f1:.4f}')
```