T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ECG CLASSIFICATION USING TRANSFORMER NETWORKS

EMRE GÜVEN

SUPERVISOR
PROF. DR. YUSUF SINAN AKGÜL

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

ECG CLASSIFICATION USING TRANSFORMER NETWORKS

EMRE GÜVEN

SUPERVISOR
PROF. DR. YUSUF SINAN AKGÜL

2024
GEBZE

GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering in 12/06/2024 by the following jury.

JURY

Member
(Supervisor) : Prof. Dr. Yusuf Sinan Akgül

Member : Dr. Yakup Genç

ABSTRACT

Electrocardiogram (ECG) signals are essential in diagnosing cardiovascular diseases (CVDs), yet their manual interpretation is labor-intensive and prone to errors. This project explores the use of Transformer Networks, a type of deep learning model known for its ability to process sequential data, to automate and improve the analysis of 12-channel ECG recordings. The goal is to enhance diagnostic accuracy and efficiency, addressing the limitations of manual interpretation.

A dataset from the online source PhysioNet, specifically the PTBXL dataset, was initially taken as the reference. However, it was imbalanced for our target three-class diagnosis. To address this, additional ECG recordings were collected, cleaned, and standardized from two other large datasets. The ECG recordings, which varied in length and sampling rates, were standardized to ensure uniformity, facilitating accurate analysis. This process included resampling the recordings to a consistent frequency and segmenting them into uniform durations, ensuring the data was ready for model training. As a result, our new dataset is larger and less imbalanced. Preliminary results using a CNN model validate the quality of the dataset and demonstrate the improvements of dataset quality.

The model development involved implementing input embeddings and positional encoding to effectively capture both local and global patterns within the ECG signals. Additional layers of the Transformer network were added to enhance the model's ability to extract significant features and dependencies from the sequential data, ultimately improving the model's diagnostic accuracy. The model was developed and trained using TensorFlow in the Google Colab environment.

Keywords: ECG, Cardiovascular Diseases, Transformer Networks, Deep Learning, Sequential Data, Positional Encoding, Data Standardization, PhysioNet, TensorFlow

ACKNOWLEDGEMENT

I would like to thank to my advisor Prof. Dr. Yusuf Sinan Akgül and teaching assistant Barış Özcan, for their valuable guidance and support throughout this project.

Emre Güven

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol / Abbreviation	Explanation
ML	Machine Learning
CNN	Convolutional Neural Network, a class of deep neural networks, most commonly applied to analyzing visual imagery.
ECG	Electrocardiogram, a test that measures the electrical activity of the heart to show whether it is working normally.
CVD	Cardiovascular disease, a class of diseases that involve the heart or blood vessels, including coronary artery diseases such as angina and myocardial infarction (heart attack).
MI	Myocardial Infarction, commonly known as a heart attack, occurs when blood flow decreases or stops to a part of the heart, causing damage to the heart muscle.
HYP	Hypertrophy, an increase in the size of an organ or tissue due to the enlargement of its component cells, often related to high blood pressure or heart conditions.
NORM	Normal or healthy, indicating the absence of disease or abnormalities in medical terms.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

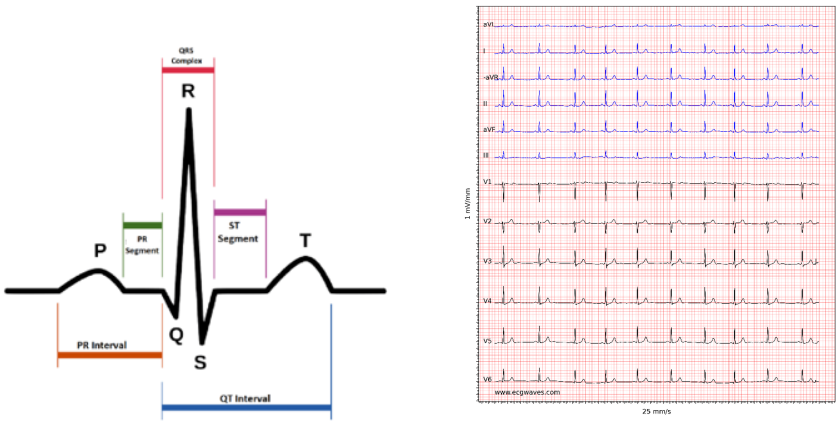
1. INTRODUCTION

In this project, a Transformer-based deep learning model was developed for the automated analysis of 12-channel ECG signals. The goal is to improve the accuracy and efficiency of CVD detection by leveraging the powerful features of Transformer Networks to process sequential data.

1.1. Project Description

This project aims to improve the diagnosis of heart diseases using Transformer Networks to analyze 12-channel ECG signals. ECG signals show the heart's electrical activity and are key to diagnosing heart conditions. Manual analysis of these signals is slow and can be inaccurate, so an automated solution is needed.

Transformers are well-suited for analyzing 12-channel ECG signals because they handle sequential data and understand long-term patterns that are essential for accurate ECG analysis. This project uses the Transformer model to automate and improve ECG analysis, making the process faster and more accurate, thus reducing the workload on doctors and enhancing patient care. Additionally, by incorporating R-peaks—important features representing the QRS complex peaks (Figure 1.1a)—the model gains valuable information about heart rhythms. This improves the model's ability to capture time-based dependencies and boosts overall classification performance in ECG signals (Figure 1.1b).



(a) QRS Complex (b) ECG Signal 12-Channels

Figure 1.1: QRS Complex and ECG Signal

1.2. Success Criteria

The success criteria for this project include creating a balanced dataset with ECG signals adjusted to 100Hz and 10-second durations. The model should achieve at least 85% classification accuracy and an F1 score of 0.8. Additionally, the model must deliver results within 10 seconds for each ECG recording. Meeting these criteria will enhance the automated and accurate analysis of ECG signals, improving cardiovascular disease diagnosis and treatment.

2. DATASET PREPARATION

This chapter outlines the detailed process of dataset preparation for the Transformer-based ECG classification model. The primary objective was to create a comprehensive, balanced, and standardized dataset from various sources, ensuring consistency in sampling rates and segment durations.

2.1. Data Collection

The dataset preparation began with the collection of ECG records from multiple datasets available on the PhysioNet website [1]. The focus was on three specific diagnoses: MI, HYP, and NORM. The primary reference dataset, the PTBXL dataset [2], comprised ECG recordings with a sampling rate of 100Hz and a duration of 10 seconds. To enhance the dataset's robustness and balance, additional ECG recordings were sourced from two other major datasets. The INCART dataset [3] included ECG recordings at a sampling rate of 257Hz with a duration of 30 minutes, while the PTB dataset [4] provided ECG recordings at a higher sampling rate of 1000Hz and a duration of 120 seconds.

Dataset	**Sampling Rate**	**Duration**	**NORM**	**MI**	**HYP**
PTB	1000Hz	120 secs	80	368	7
PTB-XL	100Hz	10 secs	9513	5148	2649
INCART	257Hz	30 mins	0	14	11

Table 2.1: ECG Recordings from Different Datasets

The details of the ECG recordings from different datasets are summarized in Table 2.1.

2.2. Data Cleaning and Standardization

To ensure uniformity and prepare the data for effective model training, the following steps were applied:

The PTB dataset records were split into 10-second segments to match the reference dataset's duration and resampled to 100Hz. Similarly, the INCART dataset was downsampled from 257Hz to 100Hz and recordings were segmented into 10-second

intervals after resampling. The PTB-XL dataset, which contains multiple diagnoses for some signals, was cleaned. To maintain consistency in sampling rates, the Fourier method was used, allowing for accurate and efficient resampling.

Following segmentation and resampling, all ECG recordings were standardized to 10-second segments with a consistent sampling rate of 100Hz.

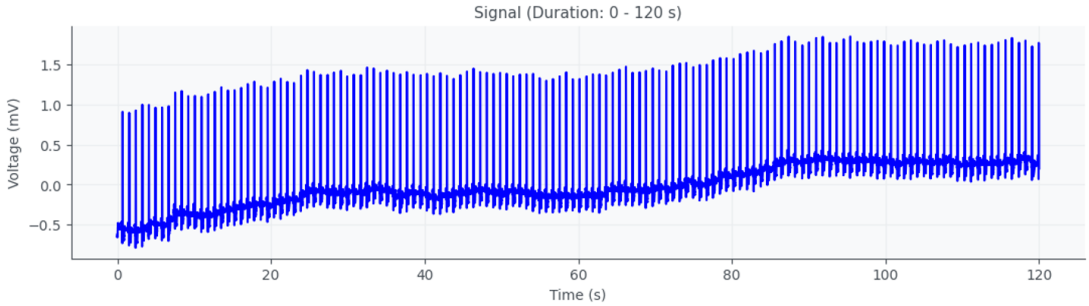


Figure 2.1: Original Long Duration Signal Channel-1

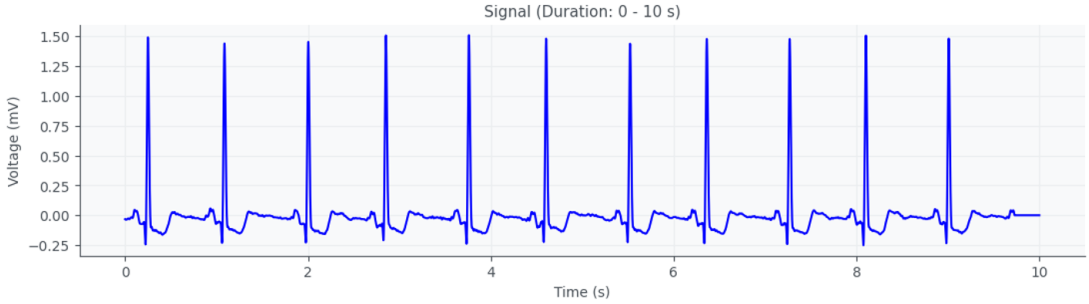


Figure 2.2: 10 secs Signal Segment Channel-1 After Processing Signal

The original long duration signal for Channel-1 is shown in Figure 2.1, and the 10-second signal segment after processing is illustrated in Figure 2.2.

Table 2.2: Distribution of the Created Dataset by Classes

	PTB		**PTB-XL**		**INCART**		**NEW TOTAL**
	Before	**After**	**Before**	**After**	**Before**	**After**	
Healthy (NORM)	80	817	9513	8577	0	0	9394
Myocardial Infarctus (MI)	368	3550	5148	3829	14	2509	9888
Hypertrophy (HYP)	7	73	2649	535	11	1982	2590

The distribution of the created dataset by classes before and after processing is shown in Table 2.2.

2.3. Data Filtering and R-Peaks Detection

Base-wander filters were applied to all ECG recordings to remove baseline wander, a common noise artifact in ECG signals. Additionally, R-peaks, which represent the peaks of the QRS complex in the ECG signals, were detected. This step is crucial as R-peaks are important features for analyzing the heart's electrical activity and improving model accuracy.

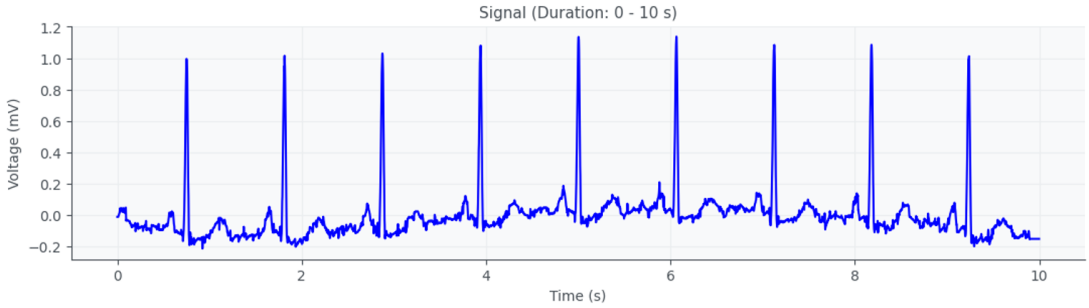


Figure 2.3: Original Signal Channel-1

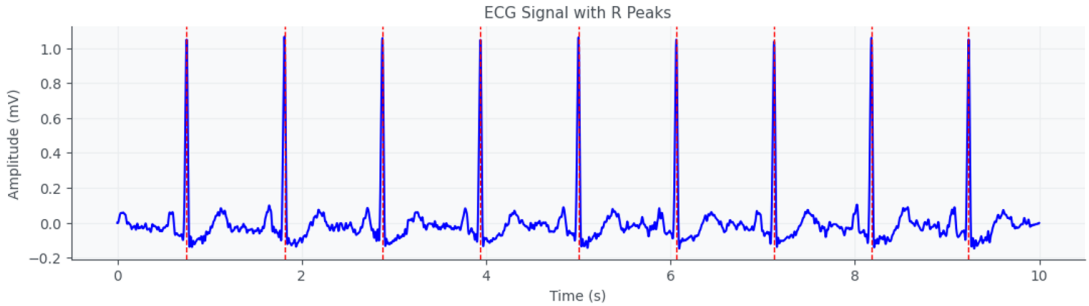


Figure 2.4: Signal Channel-1 After Base-Wander Filter and Detected R-Peaks

The original signal for Channel-1 is shown in Figure 2.3, and the signal after applying the base-wander filter and detecting R-peaks is illustrated in Figure 2.4.

2.4. Saving Data Using WFDB

The prepared data was saved using the WFDB library into three different file formats: `.dat`, `.hea`, and `.atr` [5].

The `.dat` file contains the actual ECG signal data. This file stores the ECG signal data in a binary format, holding the actual signal values that are analyzed.

The `.hea` file, also known as the header file, includes metadata about the ECG signals. Key information in this file includes the sampling frequency, the number of channels in the ECG recordings, and the total length of the ECG signals. This metadata is crucial for interpreting the signal data correctly.

The `.atr` file, or annotation file, contains the annotations for the ECG signals, specifically the R-peak points. These annotations are essential for analyzing heartbeats and detecting abnormalities in the ECG signals. The `.atr` file marks the exact locations of the R-peaks within the signal, which is critical for further analysis and interpretation.

In summary, the `.dat` file holds the raw signal data, the `.hea` file provides the necessary metadata to understand and process the signal, and the `.atr` file includes the important annotations for R-peaks, facilitating detailed analysis of the ECG recordings.

2.5. Dataset Improvement Test

This section describes the CNN model used for testing the improvement of prepared dataset.

2.5.1. CNN Test Model Overview

The model accepts input with a specific shape suitable for 1D convolution. It includes five Conv1D layers with filters of 64, 128, 256, 512, and 1024 respectively. LeakyReLU was used as the activation function for all layers. MaxPooling1D layers were applied after each convolutional layer to reduce the spatial dimensions. L2 regularization was used to prevent overfitting in all convolutional layers. Additionally, dropout layers with a rate of 0.4 were added after the dense layers to further mitigate overfitting. The output layer is a dense layer with softmax activation for classification. The model was compiled with the Adam optimizer, set to a learning rate of 0.001, and used categorical crossentropy as the loss function.

2.5.2. Test Results

The dataset is split 70% train, 20% validation, and 10% test dataset. Trainings are done with 100 epochs. The classification reports demonstrate significant improvements in model performance with the combined dataset at 100Hz compared to the original 100Hz PTBXL dataset. Notably, the F1-score for the NORM class increased from 0.8860 to 0.9048, for the MI class, it improved from 0.8385 to 0.9308, and for the HYP class, it increased from 0.4082 to 0.8544. Additionally, the overall accuracy rose from 0.8440 to 0.9104, reflecting enhanced precision, recall, and consistency across all classes. These results highlight the effectiveness of dataset enhancement in improving model accuracy and robustness. The classification report for PTBXL dataset is shown in Table 2.3, and for the combined 100Hz dataset is shown in Table 2.4.

Class	Precision	Recall	F1-Score	Support
NORM	0.9353	0.8417	0.8860	859
MI	0.7944	0.8877	0.8385	383
HYP	0.3191	0.5660	0.4082	53
Macro avg	0.6830	0.7651	0.7109	1295
Weighted avg	0.8684	0.8440	0.8524	1295
Mean F1 Score		0.7109		
Mean Precision		0.6830		
Mean Recall		0.7651		
Accuracy		0.8440 (1295 samples)		

Table 2.3: Classification Report, 100Hz PTBXL Dataset

Class	Precision	Recall	F1-Score	Support
NORM	0.8940	0.9159	0.9048	939
MI	0.9439	0.9181	0.9308	989
HYP	0.8479	0.8610	0.8544	259
Macro avg	0.8952	0.8983	0.8967	2187
Weighted avg	0.9111	0.9104	0.9106	2187
Mean F1 Score		0.8967		
Mean Precision		0.8952		
Mean Recall		0.8983		
Accuracy		0.9104 (2187 samples)		

Table 2.4: Classification Report, 100Hz Combined Dataset

3. TRANSFORMER NETWORK MODEL DEVELOPMENT

3.1. Transformer Sequence Model for ECG Classification

The Transformer sequence model is designed for the classification of ECG signals. The model processes input windows of ECG data, each containing a specified number of windows, window size, and channels. The architecture consists of several key components, described as follows:

3.1.1. Model Architecture

- **Inputs:**
  - *input_windows*: A tensor of shape (`max_windows`, `window_size`, `num_channels`) representing the segmented windows of the ECG signals.
  - *input_rpeak_windows*: A tensor of the same shape, containing R-peak information for enhanced positional encoding.
- **Embedding Layer:**
  - *TimeDistributed Conv1D Layer:* A convolutional layer with filters equal to `d_model`, kernel size of 3, and activation function specified by `activation`. This layer is applied to each window using `TimeDistributed`.
  - *Dropout Layer:* Applied after the Conv1D layer with a dropout rate of `dropout_rate`.
- **Positional Encoding:**
  - *Enhanced Positional Encoding:* Each window is passed through a lambda layer that applies enhanced positional encoding using R-peak information.
- **Transformer Encoder Blocks:**
  - *Multi-Head Attention:* Each encoder block includes a Multi-Head Attention layer with `num_heads` attention heads and a key dimension of `head_size`.

- *Feed-Forward Network (FFN):* Consists of two dense layers, the first with `ff_dim` units and the second with units equal to the input shape's last dimension. The first dense layer uses the specified `activation` function.
- *Layer Normalization:* Applied after the attention output and FFN to normalize the outputs and stabilize training.
- *Residual Connections:* Both the attention output and FFN include residual connections to preserve input information.

- **Global Average Pooling:**
  - *TimeDistributed GlobalAveragePooling1D:* Applied to each window to reduce the dimensionality by averaging the features over the time dimension.

- **Flatten Layer:**
  - *Flatten:* The output of the pooling layer is flattened to prepare for the final classification.

- **Dense Layers:**
  - *Dense Layer:* A dense layer with 108 units and the specified `activation` function.
  - *Dropout Layer:* Applied after the dense layer with a dropout rate of `dropout_rate`.
  - *Output Layer:* The final dense layer with a softmax activation function to classify the input into one of the specified classes.

- **Model Compilation:**
  - *Optimizer:* The model is compiled using the Adam optimizer.
  - *Loss Function:* Categorical cross-entropy is used as the loss function.
  - *Metrics:* Accuracy is used as the evaluation metric.

3.2. Enhanced Positional Encoding

The `EnhancedPositionalEncoding` class is a custom TensorFlow Keras layer designed to add positional encoding to the input data, while enhancing specific points, such as R-peaks in ECG signals.

3.2.1. Positional Encoding

Positional encoding is used to provide information about the relative or absolute position of the tokens in the sequence. The positional encoding is added to the input embeddings at the bottom of the encoder stack, which helps the model to understand the order of the sequence.

The positional encoding is defined as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{3.1}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right) \tag{3.2}$$

where:

- pos is the position,
- i is the dimension,
- d_{model} is the dimensionality of the model.

These equations ensure that each position has a unique encoding and that similar positions have encodings that are close to each other, which helps the model to learn the sequence information.

3.2.2. R-Peak Amplification

In addition to the standard positional encoding, the `EnhancedPositionalEncoding` class also amplifies the R-peak points in the input signal. The R-peaks are critical points in ECG signals that indicate the occurrence of heartbeats. To make these points more prominent, the R-peak annotations are multiplied by a factor, denoted as $r_{\text{peak\_factor}}$.

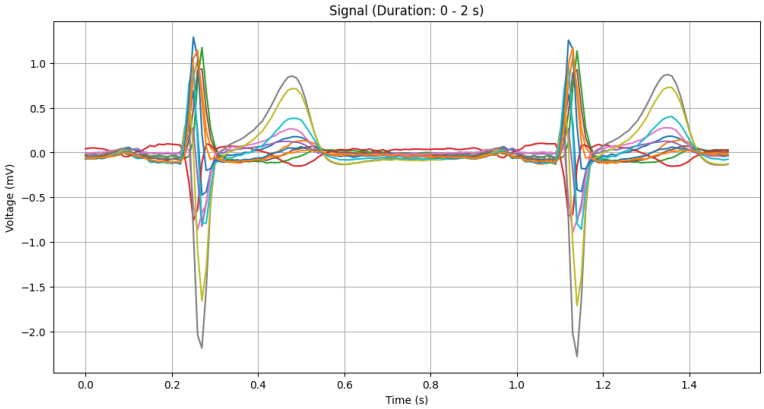
The enhanced input is computed as:

$$\text{Enhanced Input} = \text{Input} + \text{Positional Encoding} + (r_{\text{peaks}} \times r_{\text{peak\_factor}}) \tag{3.3}$$

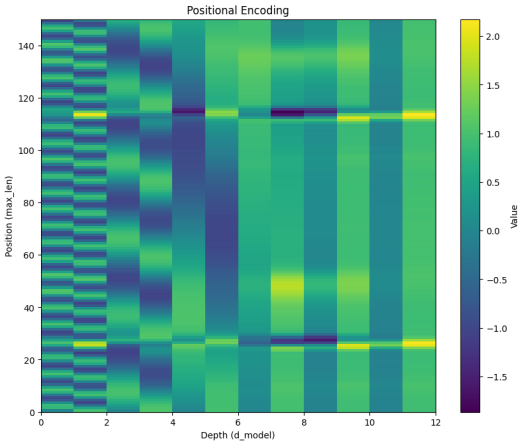
where:

- Input is the original input signal,
- Positional Encoding is the positional encoding added to the input,
- r_{peaks} is the binary annotation of R-peaks,
- $r_{\text{peak\_factor}}$ is a scaling factor to amplify the R-peaks.

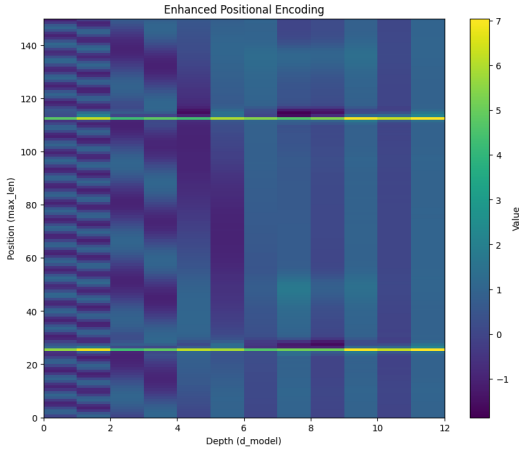
By amplifying the R-peaks, the model is better able to learn and recognize the importance of these critical points in the signal, which can improve its performance in tasks such as ECG classification.



(a) Original Window Signal



(b) Normal Positional Encoding



(c) Enhanced Positional Encoding, Enhance Factor = 5

Figure 3.1: Comparison of Original Window Signal, Normal Positional Encoding, and Enhanced Positional Encoding

As shown in Figure 3.1, the original window signal (Figure 3.1a) is compared with normal positional encoding (Figure 3.1b) and enhanced positional encoding with an enhance factor of 5 (Figure 3.1c).

3.3. Training

3.3.1. Data Pre-Processing For Training

3.3.1.1. Configuration

The configuration settings for the data pre-processing are outlined in Table 3.1.

Parameter	Value
Signal Length	1000 samples
Class Names	NORM, MI, HYP
Batch Size	8
Validation Split	20%
Test Split	10%
Window Size	200 samples
Stride	100 samples

Table 3.1: Configuration Settings for Data Pre-Processing

3.3.1.2. Data Preparation Steps

The data preparation process involves several key steps: First, list and label files by traversing through each class file (NORM, MI, HYP) to list all files and assign corresponding class labels. Second, split the dataset into training (70%), validation (20%), and test (10%) sets using stratified splitting to ensure even class distribution. Third, implement a signal generator function that reads each file's signal and annotations, extracts windows and corresponding R-peak annotations, and yields them with the class label. Fourth, create generators by defining lambda functions to generate training, validation, and test data using the signal generator function. Fifth, preprocess the data by casting the windows and R-peak annotations to appropriate data types and ensuring labels are integers. Finally, create TensorFlow datasets from the data generators, apply the preprocessing function, batch the data, and prefetch for optimal performance during training.

3.3.2. Training Configuration

The model is compiled with the following configurations: The optimizer used is the Adam optimizer with a learning rate of 0.001. The loss function employed is

Sparse Categorical Crossentropy, and the primary metric for evaluating the model's performance is accuracy.

The specific parameters for the model are as follows: The window size is specified as `window_size`, and the number of channels is `num_channels`. The maximum number of windows is denoted as `max_windows`. The model includes 3 transformer blocks. The model dimension (`d_model`) is set to 12, with 4 attention heads. The feed-forward dimension (`ff_dim`) is configured to 256. A dropout rate of 0.2 is applied to prevent overfitting. The activation function used is Leaky ReLU, and a kernel regularizer with an L2 penalty of `0.01` is employed. Additionally, the R-peak factor, which amplifies the R-peak annotations, is set to 5.

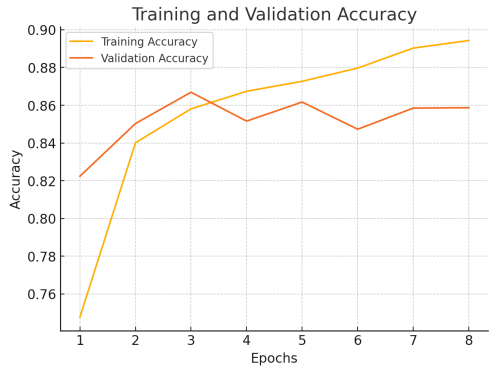
3.3.3. Training Results

This section presents the training history for two different training configurations: one with R-peak enhancement and one without. Both configurations use a window size of 150. For each training configuration, we provide the training-validation accuracy and training-validation loss graphs. Additionally, the classification results are presented.

The training-validation accuracy and loss without R-peak enhancement are shown in Figure 3.2a and Figure 3.2b, respectively, with the overall training history summarized in Figure 3.2. The corresponding classification results and evaluation metrics are detailed in Table 3.2 and Table 3.3.

Similarly, the training-validation accuracy and loss with R-peak enhancement are shown in Figure 3.3a and Figure 3.3b, respectively, with the overall training history summarized in Figure 3.3. The corresponding classification results and evaluation metrics are detailed in Table 3.4 and Table 3.5.

3.3.3.1. Training without R-Peak Enhancement



(a) Training-Validation Accuracy without R-Peak Enhancement

(b) Training-Validation Loss without R-Peak Enhancement

Figure 3.2: Training History without R-Peak Enhancement

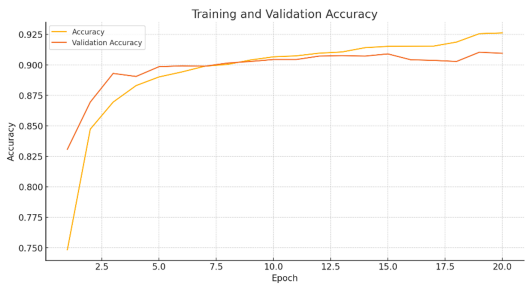
Class	Precision	Recall	F1-Score	Support
NORM	0.8235	0.9340	0.8752	939
MI	0.9207	0.8564	0.8874	989
HYP	0.9554	0.7452	0.8373	259
Accuracy			0.8765	2187
Macro Avg	0.8999	0.8452	0.8666	2187
Weighted Avg	0.8830	0.8765	0.8762	2187

Table 3.2: Classification Results without R-Peak Enhancement

Metric	Value
Mean F1 Score	0.8666
Mean Precision	0.8999
Mean Recall	0.8452
Test Loss	0.3288
Test Accuracy	0.8765

Table 3.3: Evaluation Metrics without R-Peak Enhancement

3.3.3.2. Training with R-Peak Enhancement



(a) Training-Validation Accuracy with R-Peak Enhancement

(b) Training-Validation Loss with R-Peak Enhancement

Figure 3.3: Training History with R-Peak Enhancement

Class	Precision	Recall	F1-Score	Support
NORM	0.8916	0.9201	0.9057	939
MI	0.9237	0.9060	0.9148	989
HYP	0.9153	0.8764	0.8955	259
Accuracy			0.9086	2187
Macro Avg	0.9102	0.9008	0.9053	2187
Weighted Avg	0.9089	0.9086	0.9086	2187

Table 3.4: Classification Results with R-Peak Enhancement

Metric	Value
Mean F1 Score	0.9053
Mean Precision	0.9102
Mean Recall	0.9008
Test Loss	0.2370
Test Accuracy	0.9085

Table 3.5: Evaluation Metrics with R-Peak Enhancement

4. CONCLUSION

The Transformer-based ECG classification model developed in this project incorporates enhanced positional encoding and R-peak amplification to effectively capture both local and global patterns in 12-channel ECG signals. The model architecture includes multi-head attention layers, feed-forward networks, and TimeDistributed convolutional layers, designed to process sequential data and improve diagnostic accuracy.

The results demonstrate that incorporating R-peak enhancement significantly improves the model's performance. The model with R-peak enhancement achieved a higher accuracy of 90.86% compared to 87.65% without enhancement. Furthermore, the mean F1 score increased from 0.8666 to 0.9053, indicating a more reliable performance across different classes. Precision and recall metrics also showed improvement, with the mean precision rising to 0.9102 and mean recall to 0.9008, compared to 0.8999 and 0.8452, respectively, in the non-enhanced model.

These results highlight the importance of R-peak amplification in capturing critical features of ECG signals, thereby enhancing the model's ability to detect cardiovascular diseases accurately and efficiently. The improved metrics demonstrate the potential of this approach in clinical applications, offering a more robust and reliable tool for automated ECG analysis.

BIBLIOGRAPHY

[1] A. L. Goldberger *et al.*, *Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals*, https://physionet.org, Accessed: 2024-06-12, 2000.

[2] P. Wagner *et al.*, "Ptb-xl, a large publicly available electrocardiography dataset," *Scientific Data*, vol. 7, p. 154, 2020. DOI: 10.1038/s41597-020-0495-6.

[3] L. Sidney, *The incart database*, https://physionet.org/content/incartdb/1.0.0/, Accessed: 2024-06-12, 2007.

[4] R. Bousseljot, D. Kreiseler, and A. Schnabel, *Ptb diagnostic ecg database*, https://physionet.org/content/ptbdb/1.0.0/, Accessed: 2024-06-12, 1995.

[5] A. L. Goldberger *et al.*, *Wfdb software package*, https://physionet.org/content/wfdb, Accessed: 2024-06-12, 2020.

CV

Personal Information

Address: Gebze, Kocaeli
Email: e.emreguven@outlook.com
Phone: +905326590741

Summary

A computer engineering student who works devotedly to fulfill assigned tasks, produces quality work, is willing to learn new technologies, and is prone to both individual and group work. Enthusiastic about mobile application development.

Experience

Android Developer Intern - OBSS
Developed a simple note-taking application, Pokedex application, and movie application using current technologies. Technologies: MVVM, LiveData, Coroutines, Retrofit, Dependency Injection with Hilt, Room Database, Glide.

Workshops Attended

Android Development Workshop with Advanced Kotlin - BTK Academy
Current technologies used in Android development were examined in the workshop for 6 days. MVVM architecture was implemented using sample projects.

Skills

Programming Languages: Kotlin, Flutter/Dart, C, Java, Python (for Data Science)
Technologies: Git, Android Studio, Digital Image Processing, Artificial Intelligence in Mobile Applications
Languages: English (working proficiency)

APPENDICES

GitHub Repository

You can find the application source code and model training code on GitHub:
https://github.com/eemreguven/ECG-Classification-Using-Transformer-Networks