# Comparing Recurrent Neural Networks for ECG Analysis

Sander Sæther

Thesis submitted for the degree of
Master in Applied Computer and Information Technology (ACIT)
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

OSLO METROPOLITAN UNIVERSITY

Spring 2023

# Comparing Recurrent Neural Networks for ECG Analysis

Sander Sæther

# Abstract

In this thesis, the effectiveness of three types of Recurrent Neural Networks (RNNs) - Basic RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) - are examined for Electrocardiogram (ECG) signal classification. We categorize ECG signals into five classes: normal, myocardial infarction, ST elevation, conduction disorder, and hypertrophy, with a thorough process of data preparation ensuring optimal conditions for model training and evaluation.

This study aimed to compare the performance of three types of recurrent neural networks (RNNs) - basic RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) - in classifying electrocardiogram (ECG) signals into five categories: normal (NORM), myocardial infarction (MI), ST elevation (STTC), conduction disorder (CD), and hypertrophy (HYP). The research was confined to the analysis of the PTB-XL dataset, a large publicly available ECG dataset The performance of each model is evaluated based on several metrics such as accuracy, precision, recall, and F1 score, with a separate dataset used to assess their ability to generalize. The LSTM model emerges as the most effective, demonstrating the highest accuracy (87.89%), precision, recall, and F1 score. The methodology involved preprocessing and preparation of the PTB-XL dataset, designing and implementing the RNN, LSTM, and GRU model architectures, and training, validating, and testing these models. The performance of the models was evaluated using a separate validation dataset after each training epoch and tested on a separate test dataset.

Further insights are obtained from an analysis of confusion matrices. The basic RNN model, though proficient in classifying normal ECG patterns, struggled with the classification of abnormal classes, representing a significant limitation in a clinical context. Both the LSTM and GRU models, on the other hand, were more capable in identifying abnormal cases and differentiating between classes, with the LSTM model demonstrating a more balanced performance.

In conclusion, LSTM models proved to be the most effective for ECG signal classification among the three types of RNNs evaluated. However, there is potential for further improvement and optimization, including hyperparameter tuning, ensemble methods, and the integration of other types of neural networks like Convolutional Neural Networks (CNNs). Future work could also investigate cost-sensitive learning or other methods to minimize misclassification.Overall, the adoption of advanced machine learning models, particularly LSTM, could significantly improve the speed and accuracy of cardiac disorder diagnosis, ultimately leading to better patient outcomes.

Despite LSTM's superior performance, the potential for future research to explore model optimizations and integrations with other neural network types, such as Convolutional Neural Networks (CNNs), is recognized. The study concludes by highlighting the importance of considering the high cost of misclassification in clinical settings, suggesting that future work should focus on cost-sensitive learning.

# Acknowledgments

I would like to express my sincere gratitude to my supervisors for all the guidance and support with the thesis work.

# Contents

# List of Figures

# List of Tables

# Acronyms

**ECG** Electrocardiogram. v, ix, 1–7, 37

**GRU** Gated recurrent unit. 4–7, 11, 12, 16, 17, 23, 25

**LSTM** Long Short-term Memory. 4–6, 11–13, 16, 23, 25

**RNN** Recurrent Neural Network. 1, 4–7, 10, 11, 13, 16, 23, 25, 26, 29

# Chapter 1

# Introduction

ECGs are vital tools in the diagnosis and monitoring of heart-related conditions. By capturing the electrical activity of the heart over time, these diagnostic tests provide crucial insights into the heart's functioning, enabling healthcare professionals to identify and manage various cardiac issues. As the accuracy of ECG analysis is of paramount importance, researchers have continually sought to develop and refine techniques for interpreting these complex signals. In recent years, machine learning and deep learning algorithms have emerged as promising tools for enhancing the efficiency and precision of ECG analysis. Among these, Recurrent Neural Networks (RNNs) have gained particular attention due to their ability to model sequential data. This study aims to compare the performance of different RNN architectures in analyzing ECG data, with a focus on their applicability and limitations. In this introduction, I will discuss the background and significance of ECG analysis in healthcare, provide an overview of machine learning and deep learning techniques used in ECG analysis, and introduce the role of RNNs in this context. Finally, I will outline the objectives and scope of this study and giva a summary of all the chapter.

## 1.1 Motivation

The motivation for this master's thesis is to address the limitations of existing ECG analysis methods and explore the potential of deep learning techniques in improving cardiac care. The manual interpretation of ECGs by clinicians is time-consuming and prone to errors, leading to misdiagnosis and delayed treatment. The development of automated and accurate ECG analysis methods can aid clinicians in the diagnosis and management of cardiac conditions, ultimately improving patient outcomes. LSTM RNN models have shown promising results in other time series data analysis tasks and have the ability to capture long-term dependencies in data, making them potentially suitable for ECG analysis.

## 1.2 Background on Electrocardiogram (ECG) and Its Significance in Healthcare

The ECG is a crucial, non-invasive medical tool utilized extensively in diagnosing various cardiac diseases. It measures the electrical activity generated by the heart muscle as it contracts and relaxes. The ECG has become indispensable in healthcare, playing a pivotal role in diagnosing and monitoring conditions like arrhythmias, myocardial infarctions, and heart failure [21]. ECG signals comprise distinct waveforms—P wave, QRS complex, T wave—that signify the depolarization and repolarization processes in the heart's atria and ventricles (see Figure 1.1 for a typical ECG signal) [43]. Analysis of these waveforms and their patterns can provide critical insights into the heart's functioning and potential abnormalities. Nevertheless, accurate ECG interpretation is a complex task that demands substantial expertise [10]. Even seasoned professionals may occasionally find it challenging to identify subtle patterns or anomalies within ECG data [16]. Inaccurate interpretations can result in delayed or erroneous diagnoses, potentially leading to serious repercussions for patient health outcomes [13]. Therefore, there is a significant demand for more sophisticated ECG analysis techniques that can enhance diagnostic accuracy and efficiency [2].

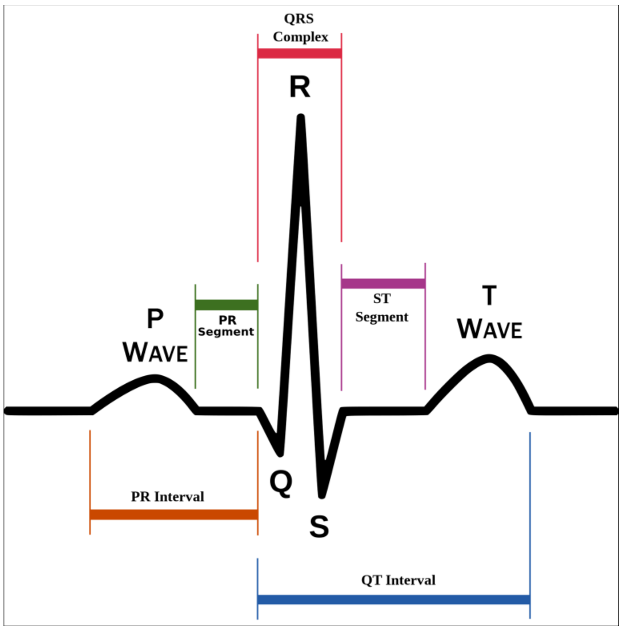

Figure 1.1: ECG signal and its features

## 1.3 Importance of Accurate ECG Analysis

The precise analysis of ECG signals plays an indispensable role in healthcare, significantly affecting patient outcomes and healthcare resource allocation. One of the primary benefits of accurate ECG analysis lies in its capacity for early detection of acute cardiac conditions. For instance, a timely diagnosed myocardial infarction (heart attack) can enable immediate interventions, significantly reducing the risk of severe complications or death [15]. Similarly, ECG analysis can facilitate the rapid identification of arrhythmias, which are often asymptomatic but can lead to serious conditions such as stroke or cardiac arrest if left untreated [35]. Beyond the acute setting, ECG analysis is also invaluable in managing chronic cardiac conditions. Proper interpretation can help track the progression of diseases like heart failure, guide treatment decisions, and monitor the effectiveness of interventions [31]. Lastly, precise ECG interpretation can enhance the efficiency of healthcare systems by reducing unnecessary diagnostic tests or treatments. Misinterpretations can lead to false-positive results, prompting additional testing, inappropriate therapies, and increased healthcare costs, not to mention the psychological distress for patients [36]. However, accurate ECG analysis can be challenging due to the complexity and variability of ECG signals. Figure 1.2, for example, shows a sample ECG plotted on a grid sheet. Subtle anomalies in these waveforms, which could indicate potential pathologies, might be difficult to discern without extensive training and experience.

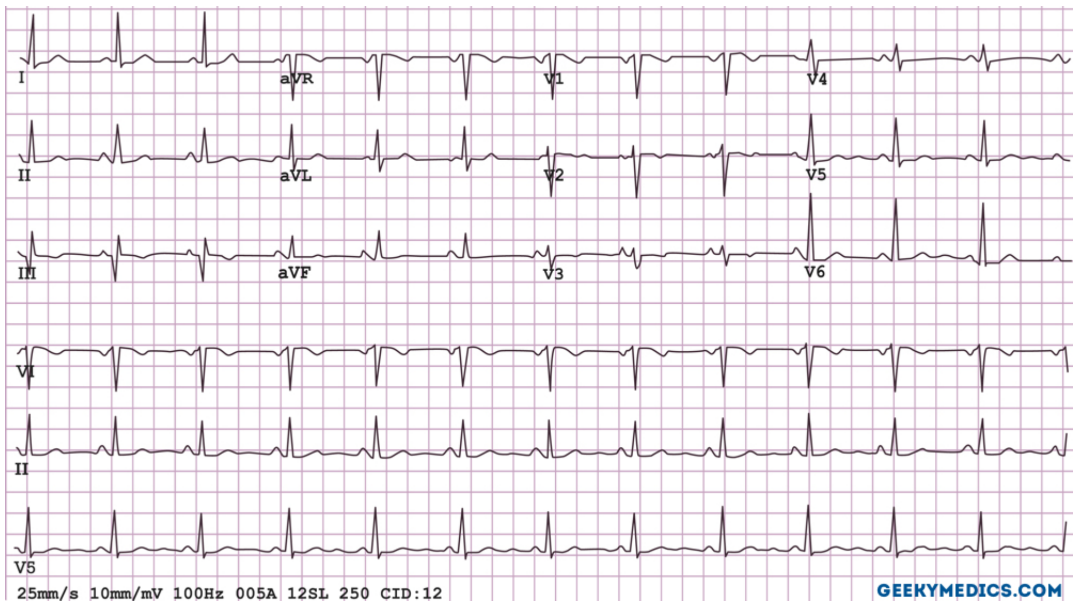

Figure 1.2: ECG sample

## 1.4 Overview of Machine Learning and Deep Learning Techniques for ECG Analysis

Over the past few decades, numerous machine learning and deep learning techniques have been developed to enhance the analysis of ECG data. Machine learning algorithms, such as support vector machines, decision trees, and k-nearest neighbors, have shown promise in detecting and classifying various cardiac conditions. However, traditional machine learning techniques often require manual feature extraction and selection, which can be time-consuming and prone to bias.

Deep learning, a subset of machine learning, has emerged as a more powerful approach to ECG analysis. Deep learning algorithms, such as convolutional neural networks (CNNs) and RNN, have demonstrated remarkable performance in various tasks, including image and speech recognition, natural language processing, and biomedical signal processing. Unlike traditional machine learning techniques, deep learning algorithms automatically learn relevant features from raw data, reducing the need for manual intervention and improving the overall accuracy and efficiency of the analysis.

## 1.5 Role of Recurrent Neural Networks in ECG Analysis

RNNs are a class of deep learning algorithms specifically designed for modeling sequential data. Unlike feedforward neural networks, RNNs possess recurrent connections, allowing them to maintain a hidden state that can store information from previous time steps. This inherent ability to capture temporal dependencies makes RNNs particularly well-suited for ECG analysis, as ECG signals are time-series data with complex temporal patterns.

Several RNN architectures have been proposed and explored in the context of ECG analysis, including vanilla RNNs, Long Short-term Memory (LSTM) networks, and Gated recurrent unit s (GRUs). LSTMs and GRUs, in particular, have shown significant potential in addressing the "vanishing gradient" problem that can affect the training of vanilla RNNs, leading to more stable and accurate models.

By leveraging the capabilities of RNNs, researchers have achieved impressive results in various ECG-related tasks, such as arrhythmia detection, myocardial infarction identification, and heart failure prediction. These advances highlight the potential of RNNs as a powerful tool for improving the accuracy and efficiency of ECG analysis.

## 1.6 Objectives and Scope of the Study

Which (RNN) architecture—LSTM, GRU, or standard RNN—provides the most effective performance in classifying ECG signals into the five main diagnostic categories within the PTB-XL dataset. The targeted ECG categories include 'NORM' (Normal), 'MI' (Myocardial Infarction), 'STTC' (ST-T Change), 'CD' (Conduction Disorder), and 'HYP' (Hypertrophy).

To address this research question, the following objectives have been set:

1. Develop and implement LSTM, GRU, and standard RNN models for ECG signal analysis, maintaining the same model architecture and hyperparameters across all models to facilitate fair comparisons.

2. Train each of the three models using the same training procedure, encompassing data preprocessing, batch size, learning rate, optimizer, and other relevant hyperparameters, to ensure that performance differences can be attributed to the model architectures rather than the training process.

3. Evaluate the performance of the LSTM, GRU, and RNN models on the test dataset using standard evaluation metrics, such as accuracy, precision, recall, and F1 score, to enable an objective comparison of the models' effectiveness in ECG analysis.

4. Analyze and discuss the strengths and weaknesses of each RNN model with respect to their ability to classify ECG signals accurately, highlighting any notable variations in model structure or behavior.

5. Determine the most suitable RNN model for ECG analysis based on the comparative evaluation of their performance and provide recommendations for future research and development in this field.

By achieving these objectives, the study aims to contribute valuable insights into the effectiveness of LSTM, GRU, and RNN models for ECG analysis, potentially guiding future research and development in this area.

## 1.7 Summary of chapters

**Chapter 2 Background**

The chapter provides an overview of machine learning techniques and their application in ECG analysis. It starts by introducing machine learning and its various subfields, such as supervised learning, unsupervised learning, reinforcement learning, feature selection, and deep learning. The focus then shifts to RNNs, specifically LSTM and GRUs, explaining their architecture, advantages, and challenges. The chapter further discusses performance metrics for evaluating neural networks, including accuracy, precision, recall, and F1-score. It also highlights the vanishing gradient problem, a significant challenge faced by traditional RNNs, and presents solutions like LSTMs and GRUs. Additionally, traditional ECG analysis methods are discussed, along with their limitations. The chapter concludes by emphasizing the potential of machine learning and deep learning approaches in ECG analysis, showcasing studies that have achieved high accuracy in ECG classification using LSTM, RNN, and GRU models.

**Chapter 3: Methodology**

this chapter presents the methodology employed in the research study to achieve the objectives outlined. The chapter begins with an overview of the data preparation process, including the use of the PTB-XL dataset for training and evaluating (RNN) models. The specific steps involved in data loading and preprocessing are discussed. The chapter then focuses on the architecture and workflow for ECG classification using RNN, LSTM, and GRU models. The data preprocessing steps common to all models are explained, followed by a detailed description of the RNN, LSTM, and GRU architectures. The training, evaluation, and prediction workflow for these models are also explained. Next, the chapter describes the procedures followed for model training, validation, and testing. The training process involves iterative updates of model weights using loss and optimization functions. The models' performance is evaluated on a validation set to monitor progress and prevent overfitting. Performance metrics such as accuracy, precision, recall, and F1 score are computed to assess the models' performance. The chapter concludes by discussing the evaluation of model performance, with a focus on the use of confusion matrices in this process. The entire methodology, from data preparation to model evaluation, is comprehensively described, providing a clear understanding of the research approach.

**Chapter 4: Results**

This chapter presents the results and analysis of three distinct RNN models: LSTM, GRU, and the basic RNN, in the classification of ECG signals from five specific categories using the PTB-XL dataset. The targeted ECG categories are 'NORM', 'MI', 'STTC', 'CD', and 'HYP'. The models' performances are evaluated using various metrics, including accuracy, F1 score, precision, and recall, on both the validation and test sets. The basic RNN model achieved an accuracy of 76.08% on the validation set, but its F1 score, precision, and recall were relatively low. This indicates that the basic RNN model struggled to achieve a good balance between precision and recall, potentially due to its inherent limitations in capturing longer-term dependencies in the ECG data. The LSTM model demonstrated improved performance, achieving an accuracy of 87.89% on the validation set. It exhibited a better balance between precision and recall, indicating its suitability for ECG classification tasks. The GRU model also showed promising results, with an accuracy of 86.84% on the validation set. When evaluated on the test set, the LSTM model maintained its superior performance, achieving an accuracy of 87.49% and an F1 score of 0.6589. The GRU model achieved an accuracy of 86.12% on the test set, demonstrating its generalization ability. The comparison of confusion matrices revealed that the basic RNN model struggled to differentiate between normal and abnormal ECG patterns, leading to misclassifications. The LSTM model showed a better balance between true positive and true negative rates across all classes, while the GRU model demonstrated high true positive rates for abnormal classes but had lower true negative rates for normal patterns. Overall, the results indicate that the LSTM model outperformed the basic RNN and GRU models in ECG classification, showing promise for the accurate identification of abnormal heart conditions. However, further exploration of hyperparameters and model refinement could lead to additional improvements in the RNN models' performance. By comparing and analyzing the performance of the three RNN architectures, this study provides valuable insights into the most effective model for ECG classification, enhancing the understanding of ECG analysis and its potential applications in clinical settings.

# Chapter 5: Discussion

This chapter highlights the results and analysis of a study comparing the effectiveness of three RNN models – RNN, LSTM, and GRU – in classifying ECG signals using the PTB-XL dataset. The LSTM and GRU models outperform the RNN model in terms of accuracy, F1 score, precision, and recall. The architectural differences between these models explain the variations in performance, with LSTM and GRU models designed to address the vanishing gradient problem inherent in traditional RNNs. And suggest some potential improvements through feature extraction, preprocessing techniques, domain knowledge integration, hybrid architectures, data augmentation, transfer learning, ensemble methods, and attention mechanisms. Researchers should consider trade-offs between model performance, complexity, and computational requirements.

# Chapter 2

# Background

Machine learning (ML) is a vibrant subfield of artificial intelligence (AI) that revolves around the creation of algorithms and methods that empower computers to learn from and make data-driven predictions or decisions [26] Its versatility has led to applications in a myriad of domains, such as natural language processing, computer vision, robotics, and biomedical signal analysis [5].

Among the various types of machine learning, supervised learning is quite prominent. In this approach, a model is trained using a labeled dataset, where each data point has a corresponding target output [5]. The primary objective is to learn a function that maps input features to output labels, enabling the model to make predictions on new, unseen data. Notable supervised learning algorithms include linear regression, logistic regression, support vector machines (SVM), and neural networks[12].

Contrastingly, unsupervised learning tackles data that lacks labeled outputs. The main goal here is to uncover hidden patterns or structures in the data, such as through clustering or dimensionality reduction [5]. Some popular unsupervised learning algorithms are k-means clustering, hierarchical clustering, and principal component analysis (PCA).

Reinforcement learning is another critical category of machine learning. It is centered around training agents to make decisions based on rewards and penalties received from their environment [41]. Reinforcement learning algorithms like Q-learning and deep Q-networks have been employed to develop agents that can master complex tasks, ranging from playing games to controlling robots, by optimizing actions to maximize cumulative rewards.

Feature selection and extraction are vital in machine learning, as they help identify and represent the most relevant information within the data [14]. Techniques for feature selection encompass filter methods, wrapper methods, and embedded methods, while feature extraction techniques include linear transformations, such as PCA, and non-linear transformations like kernel PCA or autoencoders [12].

Model evaluation is an indispensable aspect of machine learning. It allows researchers to assess their models' performance and select the most suitable one for a specific task [20]. Common metrics for evaluating classification models include accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic (ROC) curve. For regression tasks, metrics such as mean squared error, mean absolute error, and R-squared are widely used.

Deep learning, a subfield of machine learning, has garnered significant attention in recent years, owing to its success in addressing complex problems [22]. Deep learning models, particularly convolutional neural networks (CNNs) and RNNs, have exhibited remarkable performance in tasks like image classification, speech recognition, and natural language understanding.

## 2.1 Recurrent neural networks (RNN)

RNNs are a class of artificial neural networks designed to process sequential data by maintaining an internal state that captures information from previous time steps. The concept of time steps is fundamental to RNNs and represents the sequential order in which data points are processed. Each time step corresponds to one moment of time where the network receives a new input and produces an output. The sequential nature of time steps allows the network to maintain a form of "memory" by passing information from one time step to the next, thus enabling the modeling of temporal dependencies [24].

A standard RNN is composed of an input layer, one or more hidden layers, and an output layer, see figure 2.1. At each time step $t$, the input layer receives the current data point $x(t)$, and the hidden layer updates its state $h(t)$ based on both the current input $x(t)$ and the previous hidden state $h(t-1)$. This process can be described by the equation:

$$h(t) = f(h(t-1), x(t))$$

where $f$ is a non-linear activation function that defines the relationship between the current input, the previous hidden state, and the current hidden state [12]. The output layer then generates the prediction or classification for the current time step, represented by:

$$y(t) = g(h(t))$$

where $g$ is another non-linear function that maps the current hidden state to the output.

Despite the advantages of RNNs in processing sequential data, they have some limitations. One major issue faced by traditional RNNs is the vanishing

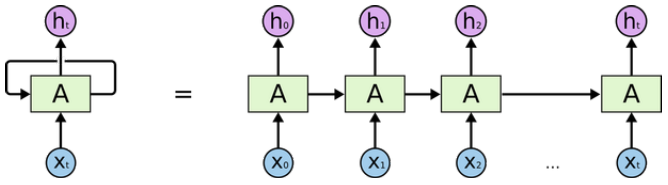

Figure 2.1: Arichitecture of basic RNN

gradient problem, which hampers their capacity to learn long-range dependencies [29]. This issue arises when the gradients of the loss function with respect to the weights become very small, causing the weights to stop updating during training. Consequently, the RNN fails to capture the relevant information from earlier time steps, limiting its effectiveness in tasks that require learning long-term dependencies.

Another challenge with RNNs is their computational complexity, especially when dealing with large datasets or long sequences. The inherently sequential nature of RNNs makes it difficult to parallelize their computations, resulting in slower training times compared to other neural network architectures, such as convolutional neural networks (CNNs) [34].

To address the vanishing gradient problem, more advanced RNN architectures have been developed, such as LSTM networks [18] and GRUs [7]. These advanced RNN models incorporate gating mechanisms that help maintain longer dependencies and overcome the vanishing gradient issue.

RNNs provide a powerful framework for modeling sequential data and capturing temporal dependencies. However, they face some limitations, such as the vanishing gradient problem and computational complexity. Advanced RNN architectures like LSTMs and GRUs have been developed to address these challenges and have demonstrated improved performance in various time-series data analysis tasks [8].

### 2.1.1 The vanishing gradient problem

The vanishing gradient problem is a significant issue that often arises during the training of artificial neural networks, particularly RNNs [29]. This problem occurs due to the nature of backpropagation, which is the algorithm used to update the network's weights during training.

Backpropagation works by calculating the gradient of the loss function with respect to the network's weights, and then using this gradient to update the weights. The gradient essentially represents the direction and magnitude of change required in the weights to reduce the loss. The weights are then updated by subtracting a fraction (determined by the learning rate) of the gradient from the current weights.

In the context of RNNs, the weight updates are performed by an algorithm known as Backpropagation Through Time (BPTT), which unrolls the recurrent network over time and applies backpropagation [12].

However, during this process, when the activation function of the network's neurons is a function like the sigmoid or the hyperbolic tangent, the gradients can become very small. This is because the derivatives of these functions are in the range of (0, 1) and (-1, 1) respectively, and when multiplied during backpropagation, can result in an exponentially decreasing product. This is mathematically represented as follows:

$$\Delta w = \eta \cdot \frac{\partial E}{\partial w} = \eta \cdot \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \text{net}} \cdot \frac{\partial \text{net}}{\partial w}$$

where $\Delta w$ is the weight update, $\eta$ is the learning rate, $E$ is the error, $y$ is the output, net is the weighted sum of inputs, and $w$ is the weight. The vanishing gradients problem occurs when $\frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial \text{net}}$ becomes very small, causing $\Delta w$ to become negligibly small. When this happens, the weights stop updating effectively, and the learning process stalls.

This problem is particularly severe in RNNs as they need to capture long-term dependencies in sequential data. When the gradient becomes very small, the RNN fails to capture the influence of past information effectively, leading to poor performance in tasks requiring long-term memory [3].

To mitigate the vanishing gradient problem, various methods have been proposed. One widely adopted approach is the use of different activation functions, such as the Rectified Linear Unit (ReLU), which does not saturate and thus reduces the likelihood of vanishing gradients. Moreover, advanced RNN architectures like LSTM [18] and GRUs [7] have gating mechanisms that allow for selective memory updates and forgetting, thus enabling the network to maintain and access information over long sequences without suffering significantly from the vanishing gradient problem.

### 2.1.2 Performance Metrics for Neural Networks

Performance metrics play an integral role in evaluating and comparing the effectiveness of neural networks across various tasks. They provide quantifiable measures that reflect the model's capability in making correct predictions and its robustness in handling different data scenarios. For classification tasks, the commonly used metrics include accuracy, precision, recall, and F1-score[39].

Accuracy is the simplest and most intuitive performance measure. It is the ratio of the number of correct predictions to the total number of input samples. It works well only if there are equal numbers of samples belonging to each class.

Precision is the ratio of correctly predicted positive observations to the total predicted positives. High precision relates to a low false-positive rate. It is also known as Positive Predictive Value (PPV).

Recall (Sensitivity or True Positive Rate) is the ratio of correctly predicted positive observations to the all observations in actual class. High recall relates to a low false-negative rate.

F1-score is the weighted average of Precision and Recall. This score tries to find the balance between precision and recall. It is a good metric to consider if there is uneven class distribution, as it seeks a balance between precision and recall[6].

### 2.1.3 Evaluating Classification Models

Classification models often use confusion matrices to evaluate their performance[39]. A confusion matrix 2.1 is a tabular representation of Actual vs Predicted values. This helps us to find the accuracy, precision, recall, and F1-score of a classification model. The four quadrants of a confusion matrix are True Positives (TP), where the positive class is correctly identified, True Negatives (TN), where the negative class is correctly identified, False Positives (FP), where the negative class is incorrectly identified as positive, and False Negatives (FN), where the positive class is incorrectly identified as negative.

Table 2.1: Confusion Matrix

| | **Predicted Positive** | **Predicted Negative** |
|---|---|---|
| Positive | True Positive (TP) | False Negative (FN) |
| Negative | False Positive (FP) | True Negative (TN) |

These values are used to calculate the metrics described above, and provide a more granular understanding of how well a model is performing beyond simple accuracy. For instance, in cases where false negatives are more damaging than false positives, models with higher recall might be favored over models with higher precision.

## 2.2 LSTM

LSTM models, introduced by Hochreiter and Schmidhuber [18], are a specific type of RNN designed to address the vanishing gradient problem. LSTMs have the ability to learn and remember long-range dependencies in time-series data, making them particularly effective for tasks involving sequences with complex temporal relationships, such as ECG analysis.

### 2.2.1 LSTM Architecture

The LSTM architecture consists of memory cells, input gates, forget gates, and output gates as illustrated in Figure 2.2. The memory cells are responsible for storing and maintaining information over long time periods. Input, forget, and output gates are used to control the flow of information into, out of, and within the memory cells.

#### Memory Cells

Memory cells are the core component of LSTM networks. Each memory cell has an internal state, which can store information over long time periods. The LSTM architecture allows for selective updates to the memory cell's internal state, preventing the vanishing gradient problem and enabling the model to learn long-range dependencies. This selective update mechanism is achieved through the use of gating units, which control the flow of information into and out of the memory cells.

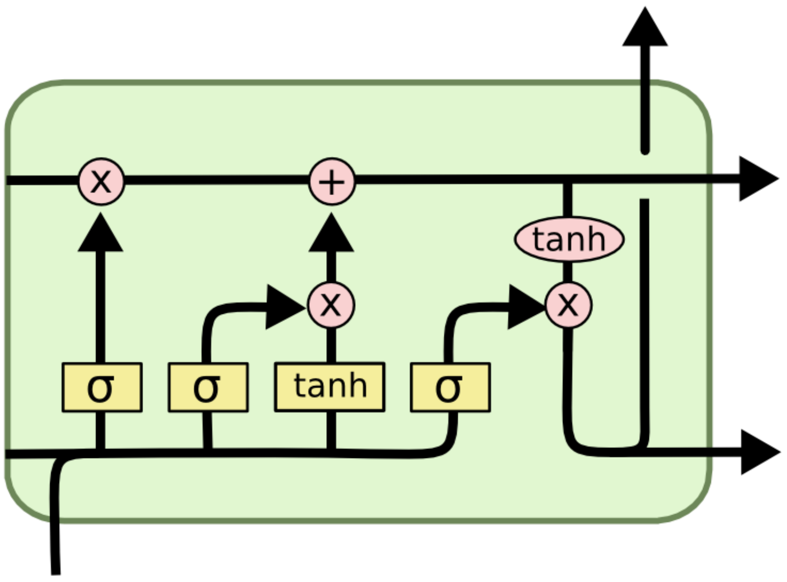

Figure 2.2: LSTM Cell diagram

#### Input Gates

Input gates manage the introduction of new information into the memory cells. They utilize a sigmoid activation function, which generates values between 0 and 1, representing the extent to which the incoming information can alter the memory cell's internal state. This gate operates in tandem with another gate that applies the hyperbolic tangent (tanh) activation function to incoming data, ensuring that input values fall within a range of -1 to 1. This gating mechanism allows the input gate to efficiently modulate the flow of information, permitting only relevant data to influence the memory cell's state. Mathematically, this can be expressed as:

$$it = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

where $it$ is the input gate's output, $\sigma$ represents the sigmoid function, $W_i$ is the weight matrix for the input gate, $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state and the current input, and $b_i$ is the bias term.

### Forget gates

Forget gates regulate the retention or removal of information from memory cells. Like the input gates, forget gates use a sigmoid activation function to decide the fraction of information to retain or discard from the memory cell's internal state. This feature allows LSTM models to "forget" irrelevant or outdated data, thereby maintaining a more accurate internal representation of time-series data. The ability to selectively remove information is especially valuable for tasks like ECG analysis, where extraneous or outdated data can impede the model's ability to identify essential patterns and make precise predictions. This can be formulated as:

$$ft = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.2)$$

Here, $ft$ is the forget gate's output, $W_f$ is the weight matrix for the forget gate, and $b_f$ is the bias term.

### Output gates

Output gates control the information flow from memory cells to the rest of the network. They utilize a sigmoid activation function to decide how much information from the memory cell's internal state should be passed to the output of the LSTM unit. The output gate also cooperates with a tanh function applied to the memory cell's internal state, ensuring the output values are within a range of -1 to 1. This mechanism enables the output gate to decide which information from the memory cell is most relevant for the current task, ensuring that only the most pertinent features are passed on to the subsequent layers of the model. This can be mathematically represented as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.3)$$

where $o_t$ is the output gate's output, $W_o$ is the weight matrix for the output gate, and $b_o$ is the bias term.

### 2.2.2 Training LSTMs

LSTMs can be trained using backpropagation through time (BPTT), similar to other RNNs. However, due to their gating mechanisms, LSTMs can effectively learn long-range dependencies without suffering from the vanishing gradient problem [18]. This property allows LSTM models to efficiently capture the temporal dynamics of ECG signals, making them a popular choice for ECG analysis tasks.

### 2.2.3 Regularization Techniques

To prevent overfitting and improve generalization in LSTM models, several regularization techniques can be applied during the training process:

#### Dropout

Dropout, introduced by Srivastava et al[40], is a technique that randomly sets a fraction of the input units to zero at each training update, preventing the model from relying too heavily on any single input feature. This method can be applied to the input, hidden, and output layers of the LSTM model, helping to reduce overfitting.

#### Weight Regularization

Weight regularization, such as L1 and L2 regularization, adds a penalty term to the loss function based on the magnitude of the model's weights. This penalty term encourages the model to learn simpler representations and reduces the risk of overfitting.

#### Gradient Clipping

Gradient clipping is a technique used to prevent the exploding gradient problem by limiting the maximum value of the gradients during backpropagation. By constraining the gradient's magnitude, gradient clipping ensures that the model's parameters are updated in a controlled manner, preventing instability during training.

## 2.3 GRU

GRUs, introduced by [7], are another type of RNN that effectively addresses the vanishing gradient problem. Similar to LSTM networks, GRUs can learn long-range dependencies in time-series data, which makes them suitable for ECG analysis tasks.

### 2.3.1 GRU Architecture

The architecture of GRUs consists of hidden states and two types of gates: update gates and reset gates as seen in figure 2.3. These gates control the flow of information within the network and are responsible for determining the retention or removal of information from the hidden states. The GRU's simplified gating mechanism, compared to LSTM, makes it computationally more efficient while still maintaining the ability to learn long-range dependencies.

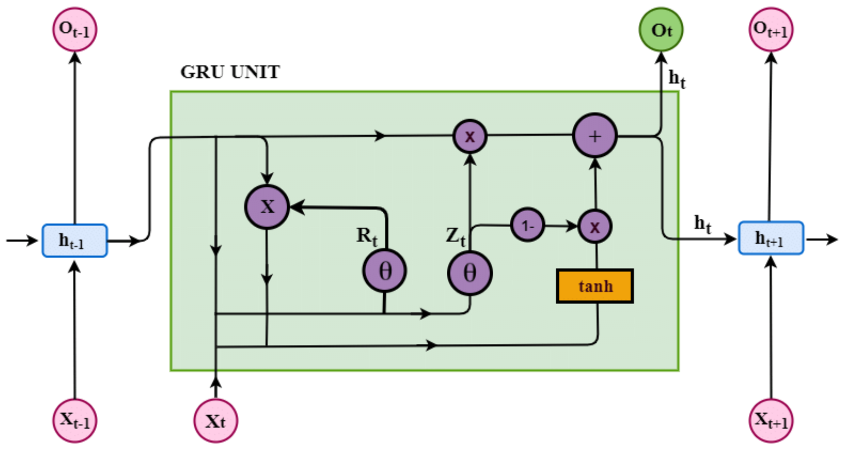

Figure 2.3: GRU architecture

#### Hidden States

Hidden states in GRUs are the primary component that allows the model to store and retain information over time. The mechanism by which hidden states are updated involves a balance between the current input and the previous hidden state, controlled by the update and reset gates. This feature enables GRUs to learn and remember long-term dependencies in time-series data, similar to LSTMs. The update rule for hidden states in a GRU can be represented as:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (2.3)$$

In this equation, $h_t$ represents the new hidden state, $z_t$ is the output of the update gate, $\odot$ denotes element-wise multiplication, $h_{t-1}$ is the previous hidden state, and $\tilde{h}_t$ represents the candidate hidden state.

#### Update Gates

Update gates in GRUs are somewhat analogous to the combined role of input and forget gates in LSTMs. They control the flow of information from the previous hidden state to the current hidden state. These gates use a sigmoid activation function to produce values between 0 and 1, which determine the proportion of information from the previous hidden state that will be retained in the current hidden state. By doing so, the update gate effectively balances the importance of past information (as represented by the previous hidden state) and the current input. This can be mathematically represented as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (2.4)$$

In this equation, $z_t$ is the output of the update gate, $\sigma$ denotes the sigmoid function, $W_z$ is the weight matrix for the update gate, $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state and the current input, and $b_z$ is the bias term.

#### Reset Gates

Reset gates in GRUs control the extent to which the previous hidden state contributes to the new hidden state. Like the update gate, the reset gate also uses a sigmoid activation function. The role of the reset gate is to decide how much of the previous hidden state will be combined with the current input to form the candidate hidden state $\tilde{h}_t$. This mechanism allows the GRU to discard irrelevant or outdated information from the hidden state while incorporating new information from the input. This selective information retention is critical for tasks such as ECG analysis, where the model needs to recognize important patterns and make accurate predictions. This can be mathematically represented as:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad (2.5)$$

In this equation, $r_t$ is the output of the reset gate, $W_r$ is the weight matrix for the reset gate, and $b_r$ is the bias term.

### 2.3.2 Training GRUs

Similar to LSTMs, GRUs can be trained using backpropagation through time (BPTT). The gating mechanisms of GRUs help them learn long-range dependencies without suffering from the vanishing gradient problem, making them effective for tasks involving complex temporal relationships such as ECG analysis.

### 2.3.3 Regularization Techniques for GRUs

To prevent overfitting and improve generalization in GRU models, regularization techniques such as dropout, weight regularization, and gradient clipping can be applied during the training process. These techniques are similar to those used for LSTMs and help in reducing overfitting and maintaining stable training.

### 2.3.4 Applications of GRUs in ECG Analysis

GRU models have been successfully applied to various tasks in ECG analysis, including arrhythmia classification, heartbeat segmentation, and anomaly detection. GRUs have demonstrated competitive performance with LSTMs in these tasks, while offering computational advantages due to their simplified gating mechanisms [8].

## 2.4 Comparison Basic RNN, LSTMs and GRUs

RNNs are a type of neural network designed for handling sequential data. They are called "recurrent" because they perform the same task for every element of a sequence, with the output being dependent on the previous computations[9]. This gives RNNs a kind of memory to capture information about what has been calculated so far.

However, RNNs have a significant problem called the vanishing gradient problem, which causes them to forget information in long sequences, making it difficult to learn long-term dependencies [3].

### 2.4.1 Basic RNNs

RNNs have an internal state that can process sequences of inputs, making them suitable for time-series data like ECG signals. However, the main limitation of a standard RNN lies in its inability to remember long-term dependencies due to issues known as vanishing or exploding gradients during training. This problem can make it hard for the RNN to learn from information that is many time steps back in the input sequence.

### 2.4.2 Long Short-Term Memory

LSTM networks, a type of RNN, were created to combat the vanishing gradient problem. They were introduced by Hochreiter  Schmidhuber in 1997 [18]. LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Using this structure, LSTMs are able to keep or delete information in the cell state over long sequences, which effectively solves the vanishing gradient problem. LSTMs are widely used and have proven to be very effective in practice, but they have a relatively complex design and can be computationally expensive.

### 2.4.3 Gated Recurrent Units

GRUs, introduced by Cho et al. in 2014 [7], are a simplified version of LSTMs. They combine the forget and input gates of an LSTM into a single "update gate". They also merge the cell state and hidden state, resulting in a simpler and more streamlined model. GRUs have been shown to perform comparably to LSTMs on certain tasks while being computationally more efficient due to their simpler structure. However, their performance relative to LSTMs can vary depending on the specific task.

### 2.4.4 Comparison

In summary, while basic RNNs are theoretically capable of handling sequential data, in practice, they struggle with long-term dependencies due to the vanishing gradient problem. LSTMs handle this issue with their more complex design involving a memory cell and gating mechanisms but at the cost of increased computational expense. GRUs offer a compromise with a simpler design that still can manage long-term dependencies more effectively than basic RNNs, but they may not perform as well as LSTMs for tasks that benefit from LSTMs' greater complexity and expressiveness. GRUs and LSTMs are both effective at learning long-range dependencies in time-series data, making them suitable for ECG analysis tasks. While LSTMs have a more complex gating mechanism with three gates (input, forget, and output gates), GRUs have only two gates (update and reset gates), making them computationally more efficient. In practice, the choice between GRUs and LSTMs often depends on the specific requirements of the task and the available computational resources.

## 2.5 Brief History of ECG Analysis Techniques

Since the invention of the electrocardiogram in the early 20th century, numerous techniques have been developed to analyze and interpret ECG signals. Early ECG analysis primarily relied on visual inspection and manual measurements, with clinicians examining the waveforms and identifying patterns that indicated abnormalities. As the field progressed, researchers began to develop more advanced methods for ECG analysis, such as mathematical and statistical approaches to quantify waveform features and detect abnormalities.

In the 1970s and 1980s, the advent of digital computers enabled the development of computerized ECG analysis algorithms. These early algorithms focused on feature extraction, pattern recognition, and classification, utilizing techniques such as Fourier analysis, wavelet transforms, and template matching. The introduction of

| Model | RNN | LSTM | GRU |
|---|---|---|---|
| **Architecture** | Simplest, with a basic loop structure | More complex, includes input, forget, and output gates | Simplified version of LSTM, combines input and forget gates into an update gate |
| **Memory** | Short-term memory due to vanishing gradient problem | Long and short-term memory, mitigates the vanishing gradient problem | Long and short-term memory, mitigates the vanishing gradient problem |
| **Computation** | Less computationally demanding due to simpler structure | Computationally intensive due to complex structure | Less computationally intensive than LSTM due to simplified structure |
| **Performance** | Can struggle with long sequences and complex patterns | Generally performs well on various tasks, including long sequences | Performance similar to LSTM on certain tasks, but may vary depending on specifics |

Table 2.2: Comparison of RNN, LSTM, and GRU Models

machine learning in the 1990s and 2000s further advanced the field, enabling more sophisticated and automated approaches to ECG analysis

## 2.6 Traditional ECG Analysis Methods and Limitations

Traditional ECG analysis methods can be broadly categorized into three groups: time-domain methods, frequency-domain methods, and time-frequency methods [23, 27, 28]. Time-domain methods involve the extraction of features directly from the ECG signal in the time domain, such as amplitude, duration, and morphology of the waveforms. Techniques in this category include QRS detection, template matching, and statistical measures [28]. Frequency-domain methods, on the other hand, focus on the analysis of ECG signals in the frequency domain, utilizing techniques such as Fourier analysis and power spectral density estimation. Time-frequency methods, such as wavelet analysis and short-time Fourier transform, combine aspects of both time and frequency domain approaches, allowing for the simultaneous analysis of ECG signals in both domains [23]. While these traditional ECG analysis methods have made significant contributions to the field, they possess several limitations. Manual interpretation of ECG data is time-consuming and labor-intensive, especially in healthcare settings with a high volume of patients. Traditional methods often struggle to capture complex and subtle patterns within ECG signals, which can lead to misinterpretations and reduced diagnostic accuracy. Traditional methods may not perform well when confronted with noisy or artifact-ridden data, as they often rely on strict assumptions about signal characteristics. ECG interpretation requires a high level of skill and experience, leading to variability in interpretation between different healthcare professionals. Human error is another significant drawback. Misinterpretation of ECG data can lead to missed diagnoses, inappropriate treatments, and potentially adverse patient outcomes. Moreover, manual ECG analysis may not be as effective in identifying subtle or complex abnormalities that a trained algorithm might pick up. Another limitation is that traditional ECG analysis focuses on specific predefined features and intervals. This approach may overlook other potentially relevant information in the ECG data. Also, the conventional analysis methods mainly consider each ECG lead independently and might miss important information that could be obtained from analyzing the interactions between the leads.

## 2.7 Machine Learning and Deep Learning Approaches in ECG Analysis

The advent of machine learning and deep learning has brought significant advances in the field of ECG analysis. These methods utilize large amounts of data and advanced algorithms to learn patterns and make predictions, making them well-suited for tasks such as ECG signal analysis where traditional methods might struggle due to the complexity of the signals. In a study conducted by Singh et al[38], the performance of LSTM, RNN, and GRU models in beat classification was investigated using the MIT-BIH Arrhythmia Database. The evaluation of these models employed various metrics such as accuracy, specificity, and sensitivity, derived from confusion matrices. The findings revealed that the LSTM model achieved an accuracy of 88.1%, surpassing the performance of the RNN and GRU models, which achieved accuracies of 85.4% and 82.5%, respectively. The study concluded that LSTM models demonstrated superior performance in the binary classification of ECG arrhythmias. Furthermore, the study suggested potential enhancements through the exploration of multi-class classification, increased epochs and neurons, as well as the utilization of Convolutional Neural Networks (CNNs). Hannun et al. (2019) developed a 1D convolutional neural network model for ECG signal classification. The model was trained on a large dataset of annotated ECG signals and was able to identify 14 different types of cardiac arrhythmias with a high degree of accuracy, showing promise for scalable and automated ECG analysis [17]. Recently, attention mechanisms, which allow models to focus on the most relevant parts of the input for making predictions, have been integrated into deep learning models for ECG analysis. For example, Yao et al. (2020) proposed an attention-based time-incremental convolutional neural network for arrhythmia detection. The model showed improved performance over traditional CNNs by effectively capturing local and global temporal dependencies in ECG signals [45]. Transfer learning, a technique where a pre-trained model is used as a starting point for a related task, has also been explored in ECG analysis. Kachuee et al. (2018) utilized transfer learning with a pre-trained CNN model for ECG arrhythmia classification, showing that this approach could effectively leverage knowledge from related tasks to improve performance [19]. In this paper [33], the authors propose a novel ECG classification algorithm designed for continuous cardiac monitoring on wearable devices with limited processing capacity. The algorithm employs a unique architecture that combines wavelet transform and multiple LSTM recurrent neural networks. The researchers found that their method outperformed previous ECG classification techniques in terms of performance. Furthermore, it was demonstrated that the proposed algorithm can meet the real-time execution requirements for continuous monitoring on wearable devices RNNs, including LSTM and GRUs, are designed specifically to handle sequential data and may be more suited to the task of ECG analysis. These models can capture long-term dependencies in the data, an aspect that is important in ECG signals where the relevance of a feature can depend on its context in the time series. There is value in exploring the application of RNNs, LSTMs, and GRUs for the same task. These models could potentially offer even better performance and uncover new insights from ECG data due to their ability to process sequential information Overall, machine learning and deep learning methods have shown significant potential in ECG analysis, offering advantages in terms of accuracy, scalability, and the ability to handle complex patterns in ECG signals. However, these methods also come with their own set of challenges, such as the need for large amounts of annotated data and the computational resources required for training and inference. Future research in this field may focus on addressing these challenges and further improving the performance of these models.

# Chapter 3

# Methodology

This chapter outlines the comprehensive methodology employed in this research to achieve the objectives of this study. We begin by discussing the data preparation process, which forms the foundation of any machine learning project. Following this, we introduce the specifics of the RNN, LSTM, GRU model architectures that were used for ECG classification. The workflow for training, validation, and testing of these models is explained, which is critical for understanding how these models learn and predict. Finally, the methodology for evaluating model performance, which included the use of confusion matrices, is discussed.

In the following sections, each of these steps will be described in detail, providing a comprehensive understanding of the methodology used in this study to compare the performance of different RNN architectures for ECG analysis. In Section 3.1, we discussed the data preparation process, which included an overview of the PTB-XL dataset used in this study and the specific steps involved in data loading and preprocessing. Section 3.2 covered the architecture and workflow for the RNN, LSTM, and GRU models used for ECG classification. We began with an overview of the data preprocessing steps common to all models, followed by a detailed description of the basic RNN , LSTM, and GRU model architectures. This section concluded with an explanation of the training, evaluation, and prediction workflow for these models In Section 3.3, we provided a detailed description of the procedures followed for model training, validation, and testing. Finally, Section 3.4 focused on the evaluation of model performance. We specifically discussed the role and interpretation of confusion matrices in this process. By the end of this chapter, a clear understanding of the complete methodology used in this research from data preparation to model evaluation should be achieved.

## 3.1 Data Preparation

In this study, we used the PTB-XL dataset[44][11], a large publicly available electrocardiography dataset, for training and evaluating our RNN models.

### 3.1.1 PTB-XL Dataset

The PTB-XL dataset is a large publicly available electrocardiography dataset containing 21799 clinical 12-lead ECG recordings sampled at either 100 Hz or 500 Hz. The dataset includes comprehensive metadata for each ECG recording, such as patient demographics, ECG waveform data, and diagnostic labels. These diagnostic labels are derived from a combination of manual annotations and automated algorithms, offering a rich and diverse set of ground truth labels for various cardiovascular conditions. See table 3.1 for information on the PTB-XL dataset

| Property | PTB-XL Dataset |
|---|---|
| Number of Samples | 21799 |
| ECG Size | 12 leads |
| Sample Rate | 500 Hz and 100 Hz |
| Number of Classes | 5 |
| Number of Sub-Classes | 71 |
| Provided Ground Truth | Yes |
| Age Range | 17 - 95 years |
| Gender Distribution | 52% Male, 48% Female |

Table 3.1: Properties of the PTB-XL Dataset

For this study, a subset of the PTB-XL dataset was utilized to perform the ECG analysis. he ECG signals in the dataset typically have a sample rate of 500 Hz. However, to reduce computational complexity while preserving crucial information in the ECG signals, the sample rate was downsampled to 100 Hz for this study. This process involves reducing the number of samples per second in each ECG signal without losing significant information, making the data more manageable for the neural network models. The PTB-XL dataset comprises various diagnostic categories or classes. The dataset has five main classes and up to 71 subclasses. However, this study focuses on the main classes, reducing the complexity of the classification task and focusing on more generalized ECG patterns. These main classes represent broad categories of cardiac conditions, providing a robust benchmark for evaluating the performance of the three RNN architectures. In figure 3.1 you can see an ECG sample from the dataset.

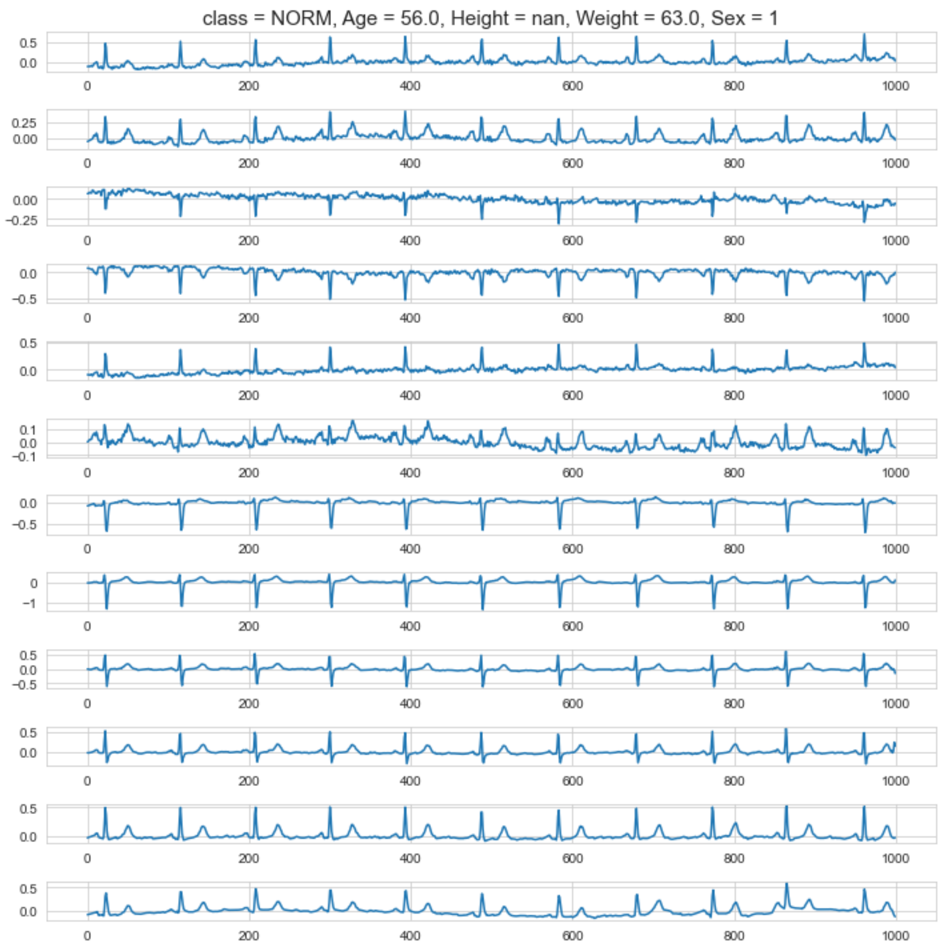
class = NORM, Age = 56.0, Height = nan, Weight = 63.0, Sex = 1

Figure 3.1: ECG sample from dataset

### 3.1.2 Data Loading and Preprocessing

The dataset was preprocessed using Python programming language and the following libraries: pandas, NumPy, wfdb, and ast. The preprocessing steps included loading the raw data, extracting the relevant information, and dividing the dataset into training, validation, and test sets. Initially, the PTB-XL dataset was loaded using the WFDB Python package, which allows for easy access to the ECG waveform data and metadata. The label file was read using Pandas, a popular data manipulation library in Python, and the scp_codes column was transformed to make it easier to work with. Next, the raw ECG signal data was loaded using the load_raw_data function, which takes the metadata DataFrame, sampling rate, and path as inputs.

| Dataset | Training Set | Validation Set | Testing Set |
|---|---|---|---|
| PTB-XL | 8 | 1 | 1 |

Table 3.2: Dataset split rates for PTB-XL

| ECG Class | % of Samples in Training | % of Samples in Validation | % of Samples in Testing |
|---|---|---|---|
| NORM | 43.6% | 43.8% | 43.7% |
| MI | 25.1% | 25.0% | 24.7% |
| STTC | 24.0% | 23.7% | 24.2% |
| HYP | 12.2% | 11.9% | 12.3% |
| CD | 22.4% | 22.6% | 22.7% |

Table 3.3: Distribution of samples in training, validation, and testing sets for each ECG class

### Diagnostic Class Aggregation

To aggregate the diagnostic labels, we first loaded the scp_statements.csv file, which contains diagnostic class information for each SCP code. A custom function, diagnostic_class, was defined to map each SCP code to its corresponding diagnostic class. The diagnostic class was then applied to the scp_codes column of the metadata DataFrame, creating a new column, target_class. Additionally, a column target_class_len was added to represent the number of diagnostic classes for each ECG recording. After aggregating the diagnostic classes, the data was divided into training, validation, and test sets using stratified sampling to ensure a balanced representation of the diagnostic classes in each set. The data was further organized by creating separate DataFrames for the ECG signal data and metadata for each partition.

### Saving Data as CSV Files

The preprocessed data were saved as CSV files to facilitate further analysis. The metadata and signal data were saved separately for each partition of the dataset. The metadata files (e.g., train_meta.csv, valid_meta.csv, and test_meta.csv) contained information such as age, sex, height, weight, nurse, site, and device, along with diagnostic class labels. The signal files (e.g., train_signal.csv, valid_signal.csv, and test_signal.csv) contained the ECG signal data for each ECG recording. The signal files were saved in a long format, where each row corresponded to a single time point in the ECG recording, and each column represented one of the 12 ECG channels. An additional column, ecg_id, was used to associate the signal data with the corresponding ECG recording in the metadata files. The ecg_id column was set as an integer to ensure proper matching between the signal and metadata files. By preprocessing the PTB-XL dataset, we ensured that the data was cleaned, organized, and ready for further analysis. This was crucial for the successful training and evaluation of the RNN models for ECG analysis. The use of stratified sampling for partitioning the dataset enabled us to maintain a balanced distribution of diagnostic classes across the training, validation, and test sets, increasing the reliability of the performance metrics computed during the model evaluation stage. Additionally, the separate storage of metadata and signal data allowed for efficient data handling during the model training process, reducing the memory overhead and ensuring smooth execution of the training and evaluation pipelines.

**PTBXLDatasetPreprocesser**

The PTBXLDatasetPreprocesser class is a vital component of the code, responsible for preparing the PTB-XL dataset for training and evaluation of the different RNN models. It streamlines the preprocessing pipeline by handling missing values, scaling numeric features, and encoding categorical features, ensuring that the data is in a suitable format for model training and evaluation. This class contains several methods to manage various aspects of data processing and transformation. The main methods include: fit(): It learns the preprocessing parameters from the training data. It identifies the class columns, handles missing values in the meta numeric columns, fits the min-max scaler, and fits label encoders to the meta categorical columns. transform(): This method applies the preprocessing pipeline to the given data. It reshapes the input ECG signals, applies the min-max scaling to the meta numeric columns, transforms the meta categorical columns using the fitted label encoders, and processes the class targets. Additionally, the class provides save() and load() methods to save and load the preprocessed data, including the learned preprocessing parameters. In summary, the PTBXLDatasetPreprocesser class efficiently handles the data preprocessing tasks, enabling the smooth training and evaluation of the RNN models on the PTB-XL dataset.

**ECGDataset**

The ECGDataset class is a custom dataset class that inherits from PyTorch's Dataset class, enabling seamless integration with PyTorch's data loading utilities. It is designed to handle the ECG signals and associated metadata for model training and evaluation. This class takes the following inputs during initialization:

- signals: A NumPy array or tensor containing the ECG signal data.
- num_metas: A DataFrame containing the numerical metadata.
- cat_metas: A DataFrame containing the categorical metadata.

- class_labels: A DataFrame containing the class labels for the ECG signals.

## 3.2 Model Architectures and Workflow for ECG Classification

Here is the logical flow for each of your three models, RNN, LSTM, and GRU, for ECG classification using the PTB-XL dataset. Figure 3.2 shows the model architecture for my model.

### 3.2.1 Data preprocessing for all models

The ECG data, along with metadata, is preprocessed before being fed into the models. This may involve normalization, filtering, or other transformations to make the data suitable for training and testing. The data is then split into training and testing sets and organized into a format that can be used by the models (i.e., PyTorch DataLoader).

### 3.2.2 Basic RNNmodel Architecture

The RNN model is a Recurrent Neural Network that can handle sequential data like ECG signals. The model has multiple layers, including an RNN layer for the ECG signal data, a fully connected layer for numerical metadata, and an embedding layer for categorical metadata. These layers are combined and passed through additional fully connected layers before the final output layer, which uses a sigmoid activation function to produce the probability of each class.

### 3.2.3 LSTM model architecture

The LSTM model is a Long Short-Term Memory network, a variant of the RNN specifically designed to handle long-term dependencies in sequential data like ECG signals. The model has multiple layers, including an LSTM layer for the ECG signal data, a fully connected layer for numerical metadata, and an embedding layer for categorical metadata. These layers are combined and passed through additional fully connected layers before the final output layer, which uses a sigmoid activation function to produce the probability of each class.

### 3.2.4 GRU model architecture

The GRU model is a Gated Recurrent Unit network, another RNN variant designed to handle long-term dependencies in sequential data like ECG signals. The model has multiple layers, including a GRU layer for the ECG signal data, a fully connected layer for numerical metadata, and an embedding layer for categorical metadata. These layers are combined and passed through additional fully connected layers before the final output layer, which uses a sigmoid activation function to produce the probability of each class.

### 3.2.5 Training, evaluation and prediction

Each model is trained separately using the training set, with loss and optimization functions being used to update the model weights iteratively. During training, the model's performance is periodically evaluated on the validation set to monitor progress and adjust the model if necessary. This process continues until the model converges or a specified number of epochs is reached. The use of the validation set helps in preventing overfitting and selecting the best model iteration during the training process. Each trained model is evaluated on the testing set. Performance metrics such as accuracy, F1 score, precision, and recall are calculated to assess the model's performance. The trained models can be used to make predictions on new, unseen ECG data by inputting the signal, numerical metadata, and categorical metadata into the model.

## 3.3 Training, validation and testing

### 3.3.1 Training and validation

The LSTM, RNN, and GRU models are trained for 20 epochs on a computing system comprising of an appropriate configuration of CPUs and GPUs, with adequate RAM to handle the data and the models. This is enabled by the use of PyTorch, a popular deep learning framework. The training data comprises ECG signal data, numerical metadata, categorical metadata, and class labels. These are prepared for input into the models through a custom function prepare_dataloader, that converts them into a format suitable for input to deep learning models. The models are implemented in PyTorch, taking advantage of its easy-to-use and efficient computation capabilities. Each ECG signal is processed within a reasonable time frame, allowing for speedy training and validation of the models. The models are optimized using the Adam optimizer, which implements a variant of gradient descent optimization algorithm with momentum. The learning rate for the optimizer 1e-2, see figure 3.4 for all hyperparameters. Other parameters of the optimizer are kept as default, as provided by the PyTorch framework. The models are trained for 20 epochs, with each epoch consisting of a forward pass and a backward pass of all training examples. The models' weights are updated in each epoch to minimize the binary cross-entropy loss. A learning rate scheduler is employed, which adjusts the learning rate after a predetermined number of epochs, to fine-tune the learning process. The performance of the models is evaluated on a separate validation dataset after each training epoch. This provides an unbiased estimate of the model's performance and its ability to generalize to unseen data. Several metrics, including accuracy, precision, recall, and F1 score, are computed for each epoch to monitor the model's performance. Upon completion of training, the models' weights are saved for future use or deployment. To ensure efficient use of resources, the models and other large objects are deleted, and the GPU memory is cleared. Throughout the training process, the wandb library is used to log important metrics and monitor the performance of the models in real-time. This tool allows for efficient tracking of the models' training and validation progress, offering an easy way to visualize the performance metrics.

### 3.3.2 Testing

Testing the Models: Upon training completion, the models' performance was evaluated using a separate test dataset. This test data was preprocessed in the same manner as the training and validation data, ensuring consistency across all sets. The test data was loaded using a data loader function, prepare_dataloader, which handled the batching and shuffling of the data. The trained models, LSTM, GRU, and RNN, were each loaded into the testing environment with their respective parameters. In each iteration of the test data loader, the ECG signals and associated metadata were passed through the model, and the predicted output was collected. As in the training phase, the signals and metadata were converted to the appropriate data types and moved to the correct device before being passed into the model. The model output was then compared to the actual labels to compute the performance metrics. These metrics included accuracy, precision, F1 score, and recall, all of which provide a comprehensive understanding of the model's performance. The predictions were logged using the Weights  Biases tool (WandB) for visualization and tracking purposes. The function predict returned the predicted probabilities and actual labels, allowing for further analysis of the model's performance. The predicted probabilities were then converted into binary predictions, which were used to calculate the aforementioned performance metrics. The testing process was designed to evaluate the performance of the trained models on unseen data. This is critical in assessing the models' generalizability, which is a measure of how well they can predict outcomes for new, unseen data based on what they have learned during training."

| Hyperparameter | Value |
|---|---|
| Learning Rate | 1e-2 |
| Step Size (for Learning Rate Scheduler) | 2 |
| Batch Size | 128 |
| Epochs | 20 |
| Hidden Size | 128 |
| Embedding Size | 30 |
| Dense Layer size | 128 |
| Number of GRU, LSTM, RNN Layers(only 1 type) | 1 |
| output size | 5 |

Table 3.4: Hyperparameters used in the model training

## 3.4 Evaluation

To assess the performance of the RNN, LSTM, and GRU models in ECG classification tasks, I employed confusion matrices as the evaluation metric. This metric enables us to gain a comprehensive understanding of each model's performance and to compare their effectiveness in accurately classifying ECG signals.

### 3.4.1 confusion matrix

A confusion matrix is a table that represents the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions for each class, providing a more detailed analysis of the model's performance across different classes. In our study, we calculated confusion matrices for each class in the PTB-XL dataset.

To generate the confusion matrices, we used the Scikit-learn library's confusion_matrix function. For each class, we calculated the confusion matrix and normalized the values to represent the proportions of the total number of instances for each true class. Using Matplotlib, we generated a visual representation of the matrices in a 2x3 grid, displaying the normalized values for each cell in the matrices.

By analyzing the confusion matrices, we can identify the strengths and weaknesses of each model in differentiating between classes. We can also determine the instances where the models might have confused one class for another, leading to misclassifications. This information is valuable for understanding the limitations of the models and can help inform future improvements.

Moreover, confusion matrices provide insights into the models' sensitivity and specificity, which are crucial metrics for evaluating classification performance, especially in medical applications. Sensitivity, also known as recall, is the proportion of true positive instances among all actual positive instances. It measures the ability of the model to correctly identify positive instances. Specificity, on the other hand, is the proportion of true negative instances among all actual negative instances. It measures the ability of the model to correctly identify negative instances.

By employing confusion matrices as our evaluation metric, we can comprehensively assess the performance of our RNN, LSTM, and GRU models in ECG classification tasks and compare their effectiveness in classifying ECG signals accurately. The insights gained from the confusion matrices can guide us in refining the models, optimizing their hyperparameters, and exploring alternative approaches to improve their performance in future studies.

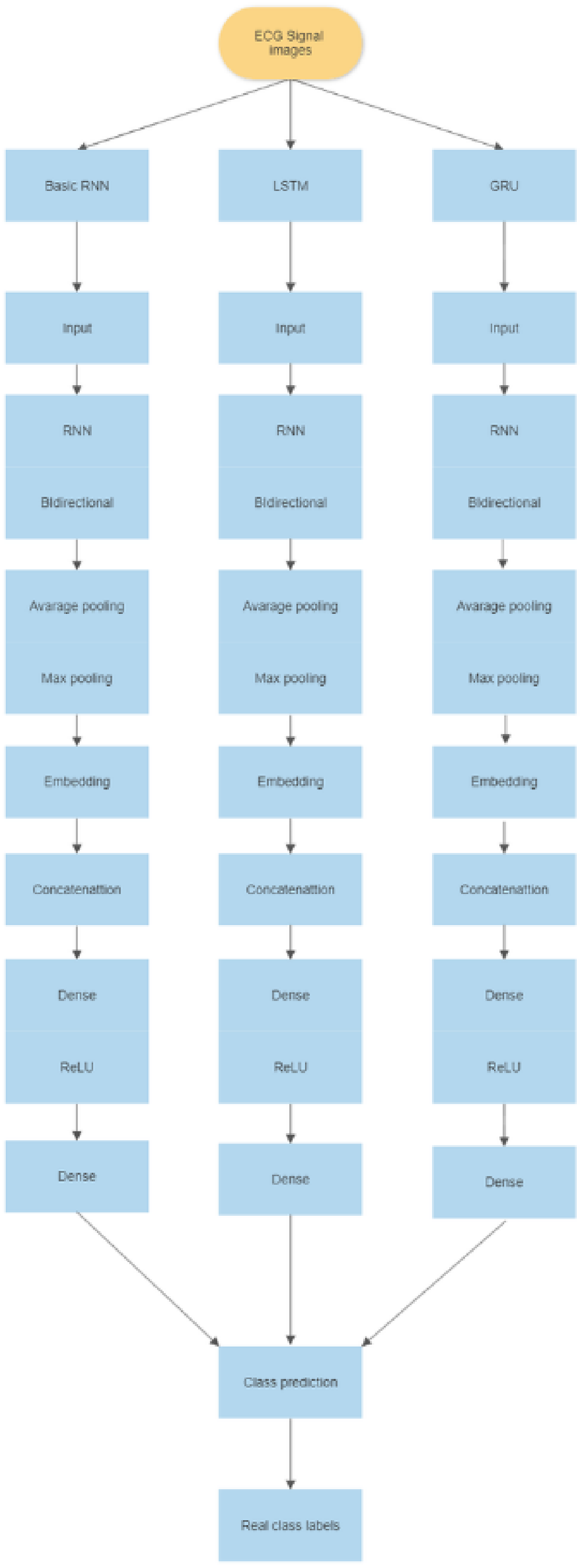

Figure 3.2: architecture

# Chapter 4

# Results

The primary objective of this master's thesis is to investigate the effectiveness of Recurrent Neural Network (RNN) models in classifying ECG signals from five specific categories using the PTB-XL dataset. The targeted ECG categories include 'NORM' (Normal), 'MI' (Myocardial Infarction), 'STTC' (ST-T Change), 'CD' (Conduction Disorder), and 'HYP' (Hypertrophy). This chapter presents the results obtained from training and evaluating three distinct RNN architectures: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and the basic RNN. The models' performances are compared and analyzed to determine the most suitable RNN architecture for ECG classification tasks focusing on these five categories. The training process was carried out for 20 epochs. To evaluate the model during training, performance on the validation set was analyzed using various metrics such as accuracy, F1 score, precision, and recall.

In this chapter, the results for each RNN model are presented separately, starting with the RNN model, followed by the LSTM model, and finally the GRU model. Each model's performance is assessed using various metrics, including accuracy, F1 score, precision, and recall, on both the training and test sets. The results are then analyzed to determine the most effective RNN architecture for ECG classification, taking into consideration the strengths and weaknesses of each model and their ability to classify the five targeted ECG categories.

| Model | Accuracy | precision | recall | f1 score |
|---|---|---|---|---|
| RNN | 0.7608 | 0.01249 | 0.07079 | 0.09286 |
| GRU | 0.8684 | 0.8228 | 0.5514 | 0.6478 |
| LSTM | 0.8789 | 0.835 | 0.5833 | 0.6754 |

Table 4.1: Summary of the results on the RNN, GRU, and LSTM models under training on validation set

| Model | Accuracy | precision | recall | f1 score |
|---|---|---|---|---|
| RNN | 0.7608 | 0.01249 | 0.07079 | 0.09286 |
| GRU | 0.8612 | 0.8202 | 0.5198 | 0.62 |
| LSTM | 0.8749 | 0.8247 | 0.5663 | 0.6589 |

Table 4.2: Summary of the results on the RNN, GRU, and LSTM models on the test set

## 4.1 Model Performance on validation Set and test set

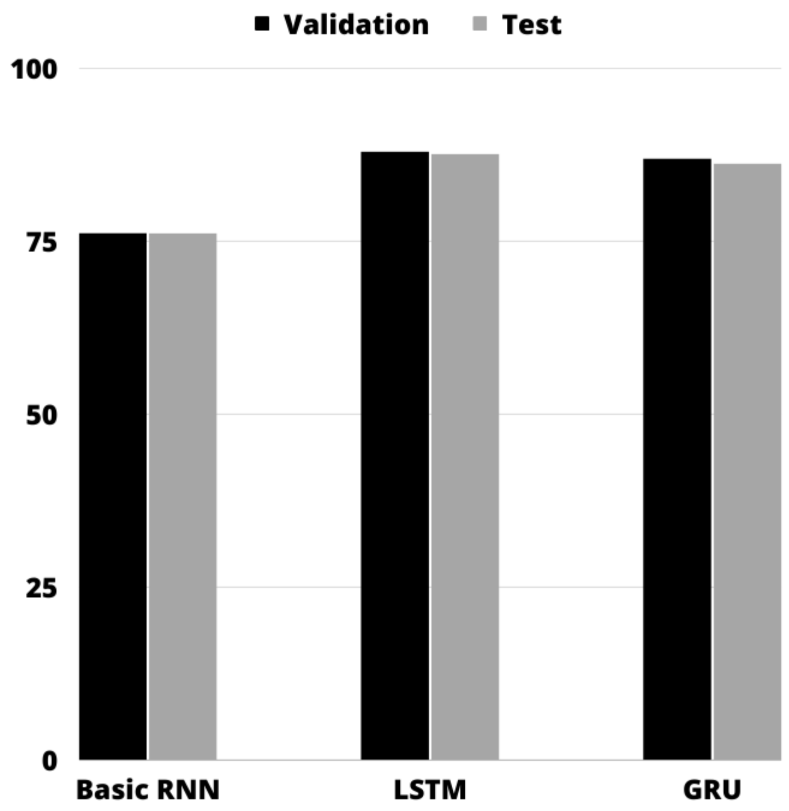

Figure 4.1: Accuracy of models on validation and test set

### 4.1.1 Basic RNN

#### Results on validation set

During the training process, the RNN model achieved an accuracy of 0.7608 on the validation set. This indicates that the model correctly classified approximately 76.08% of the samples across the five ECG categories. However, the model's performance in terms of F1 score, precision, and recall was less impressive. It obtained an F1 score of 0.09286, a precision of 0.01249, and a recall of 0.07079. These results suggest that the RNN model struggled to maintain a good balance between precision and recall, making it less suitable for ECG classification tasks compared to the LSTM and GRU models.

One possible explanation for the RNN model's lower performance compared to the LSTM and GRU models is its architecture's inherent limitations. The basic RNN architecture is prone to the vanishing gradient problem, which can hinder the model's ability to learn longer-term dependencies in the ECG data. This shortcoming could negatively impact the model's capacity to classify ECG signals effectively.

#### Results on test set

When tested on the unseen test set, the RNN model yielded an accuracy of 0.758, which is consistent with its performance on the validation set during training. However, the model's F1 score, precision, and recall remained low, with an F1 score of 0.08137, a precision of 0.1301, and a recall of 0.05919. These results reinforce the notion that the RNN model may not be the most appropriate choice for ECG classification, particularly when compared to the LSTM and GRU models.

The consistent performance of the RNN model on both the validation and test sets suggests that the model is not overfitting the training data, but it still lacks the ability to generalize well to new data. This could be due to the limited capacity of the basic RNN architecture to capture complex patterns and features in the ECG signals.

#### Analysis and Discussion

The results of the RNN model reveal some limitations in its ability to classify the five ECG categories in the PTB-XL dataset. While the model's accuracy appears acceptable, the low F1 score, precision, and recall values indicate that it struggles to achieve a balance between identifying true positive cases and avoiding false positive predictions. This may be due to the inherent shortcomings of the RNN architecture, which is susceptible to the vanishing gradient problem and may have difficulties learning longer-term dependencies in the ECG data. In comparison, the LSTM and GRU models demonstrated better performance in terms of accuracy, F1 score, precision, and recall, indicating that they are more suitable for ECG classification tasks. The improved performance of these models can be attributed to their unique architectures, which are specifically designed to overcome the limitations of traditional RNNs. The choice of hyperparameters and the training process could also have influenced the RNN model's performance. For instance, exploring alternative learning rates, different optimization algorithms, or varying the number of hidden layers and units could potentially lead to improvements in the model's performance.

Further experimentation with these hyperparameters might reveal opportunities for enhancing the RNN model's effectiveness in ECG classification tasks.

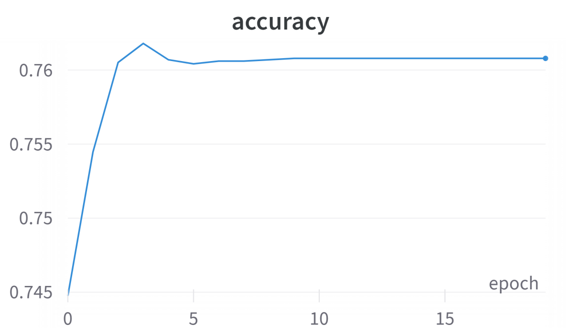

Figure 4.2: RNN accuracy on validation set under training

### 4.1.2 LSTM

#### Results on validation set

The LSTM model achieved an accuracy of 0.8789 on the validation set during training, indicating that it was able to correctly classify approximately 87.89% of the samples across the five ECG categories. Additionally, the model achieved an F1 score of 0.6754, a precision of 0.835, and a recall of 0.5833. These metrics suggest that the LSTM model provides a good balance between precision and recall, making it a strong contender for ECG classification tasks.

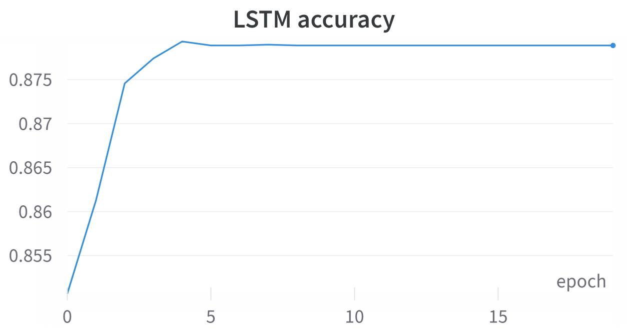

Figure 4.3: LSTM accuracy on validation set under training

#### Results on test set

When evaluating the LSTM model on the test set, it achieved an accuracy of 0.8749, which is consistent with its validation performance during training. The model obtained an F1 score of 0.6589, a precision of 0.8247, and a recall of 0.5663. These results indicate that the model generalizes well to unseen data, maintaining a good balance between precision and recall across the five ECG categories.

**Analysis and Discussion**

The results of the LSTM model indicate its effectiveness in classifying the five ECG categories in the PTB-XL dataset. When compared to the RNN model, the LSTM model shows a marked improvement in accuracy and F1 scores. Furthermore, it slightly outperforms the GRU model in terms of these metrics. The LSTM's architecture is specifically designed to address the vanishing gradient problem and learn longer-term dependencies in sequential data, which may have contributed to its superior performance in ECG classification.

The higher precision achieved by the LSTM model suggests that it can make fewer false positive predictions, while its improved recall indicates a better ability to identify true positive cases. This improved performance is especially important in the context of ECG classification, as accurate detection of abnormal heartbeats and conditions is crucial for effective diagnosis and treatment. The confusion matrix highlights the model's performance disparities between various classes, which can be attributed to several factors. One possibility is class imbalance, where the model may be biased towards the majority class and underperform in recognizing minority classes. In this case, researchers can explore techniques such as resampling or using class-weighted loss functions to address the imbalance and improve model performance. Another factor that might influence the model's performance is the quality of features extracted from the ECG data. Researchers can experiment with different preprocessing techniques or feature extraction methods, which can impact the model's ability to learn and generalize from the data. Incorporating advanced feature selection methods or employing domain-specific knowledge to extract clinically relevant features might contribute to enhanced model performance. Several factors could explain the observed performance of the LSTM model. First, the unique architecture of LSTMs may have played a significant role in the model's ability to learn the underlying patterns and features in the ECG data. The LSTM's ability to maintain long-term dependencies and avoid the vanishing gradient problem sets it apart from traditional RNNs and may have contributed to its better performance in ECG classification.

Second, the choice of hyperparameters and the training process could also have influenced the LSTM model's performance. For instance, the selection of an appropriate learning rate, the size of the hidden layers, and the optimization algorithm could all impact the model's ability to learn and generalize from the data. Further exploration of these factors may reveal opportunities for additional improvements in the LSTM model's performance.

Lastly, it is important to consider the possibility of trade-offs between the LSTM and GRU models. While the LSTM model demonstrated slightly better performance in terms of accuracy and F1 scores, it may come at the cost of increased complexity and computational requirements. Evaluating these trade-offs is crucial in determining the most appropriate model for ECG classification tasks and can help inform the selection of the best method for ECG signal analysis.

### 4.1.3 GRU

#### Results on validation set

The GRU model achieved an accuracy of 0.8684 on the validation set during training, indicating that it was able to correctly classify approximately 86.84% of the samples across the five ECG categories. Additionally, the model achieved an F1 score of 0.6478, a precision of 0.8228, and a recall of 0.5514. These metrics suggest that the GRU model has a good balance between precision and recall, providing a more effective ECG classification compared to the RNN model.

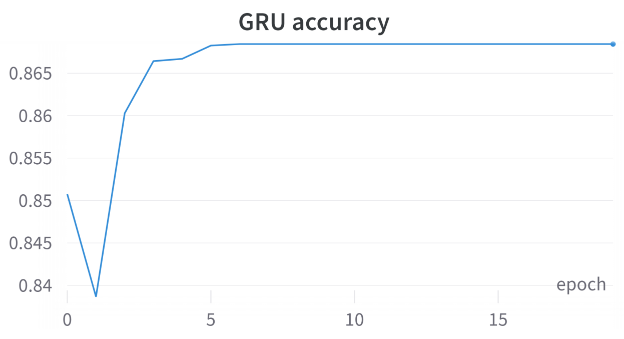

Figure 4.4: GRU accuracy on validation set under training

#### Results on test set

When evaluating the GRU model on the test set, it achieved an accuracy of 0.8612, which is consistent with its training performance. The model obtained an F1 score of 0.62, a precision of 0.8202, and a recall of 0.5198. These results indicate that the model generalizes well to unseen data, maintaining a good balance between precision and recall across the five ECG categories.

**Analysis and Discussion**

The results of the GRU model suggest that it is effective in classifying the five ECG categories in the PTB-XL dataset, achieving higher accuracy and F1 scores compared to the RNN model. The improved performance can be attributed to the GRU's architecture, which is designed to address the vanishing gradient problem commonly encountered in RNNs. This allows the GRU model to learn longer-term dependencies in the ECG data, leading to more accurate classification. The higher precision of the GRU model indicates that it is capable of making fewer false positive predictions, while the improved recall suggests a better ability to identify true positive cases. This performance is particularly important in the context of ECG classification, where accurate identification of abnormal heartbeats and conditions is critical for effective diagnosis and treatment.

It is essential to understand the reasons behind the observed performance of the GRU model. The architecture of GRUs may have contributed significantly to the model's ability to learn the underlying patterns and features in the ECG data. Furthermore, the choice of hyperparameters and training process could also have influenced the performance, and further exploration of these factors may reveal opportunities for additional improvements.

## 4.2 Comparison of confusion matrices

The confusion matrices for the RNN, LSTM, and GRU models provide insights into their performance in classifying ECG signals across different classes (see Figures 4.5, 4.6, and 4.7 for the respective confusion matrices). A detailed comparison of the three models is discussed below.

### 4.2.1 Basic RNN model

The RNN model demonstrates a high true positive rate for the NORM class at 88%, indicating that it can accurately identify normal ECG patterns in most cases. However, it struggles with the classification of abnormal cases, as the true positive rate for MI, STTC, CD, and HYP is 100%. This means that the model misclassifies all instances of these classes as normal, which is a significant limitation, see figure4.5.

The false positive rate for the NORM class is 12%, whereas it is 0% for the remaining classes. The true negative rate for the NORM class is 30%, which is considerably lower compared to the LSTM model. This implies that the RNN model has difficulty differentiating normal ECG patterns from abnormal cases, leading to a

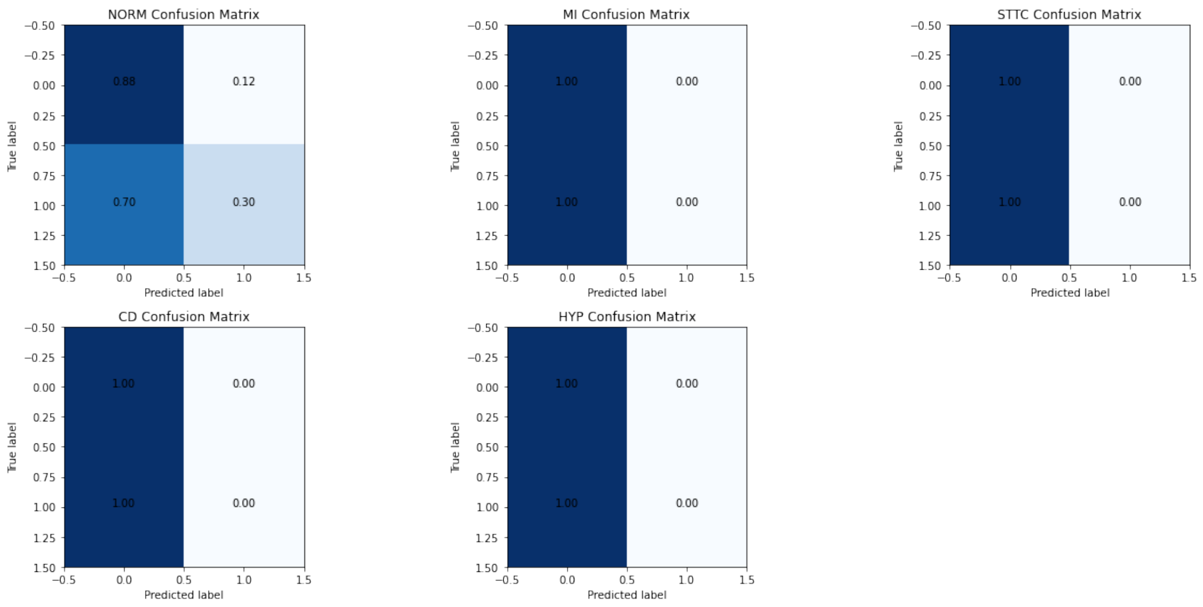

Figure 4.5: confusion matrix RNN

higher rate of false positives.

Furthermore, the false negative rates for the MI, STTC, CD, and HYP classes are all 100%, suggesting that the RNN model is unable to accurately identify any abnormal cases within these classes. This highlights a major drawback of the RNN model, as it fails to recognize critical abnormalities in the ECG data, which could have significant implications in clinical settings.

In addition, the true negative rates for MI, STTC, CD, and HYP are all 0%, further emphasizing the model's inability to distinguish between different abnormal classes. This is in stark contrast to the LSTM model, which demonstrates a better balance between true positive and true negative rates across all classes.

### 4.2.2 LSTM Model

The confusion matrix for the LSTM model reveals valuable insights into the model's performance in ECG classification across different classes. The model demonstrates high true positive rates for all classes, with values ranging from 87% for NORM to 99% for HYP. This indicates that the model is successful in detecting abnormal ECG patterns and classifying them correctly. Additionally, the false positive rates are relatively low, ranging from 1% for HYP to 13% for NORM, see figure4.6. However, the true negative rates vary across classes, with values such as 84% for NORM, 55% for MI, 57% for STTC, 56% for CD, and a notably low 31% for HYP. These values suggest that the model might struggle to recognize normal ECG patterns or differentiate between specific abnormal cases, leading to misclassification. Despite these challenges, the LSTM model shows a more balanced performance compared to

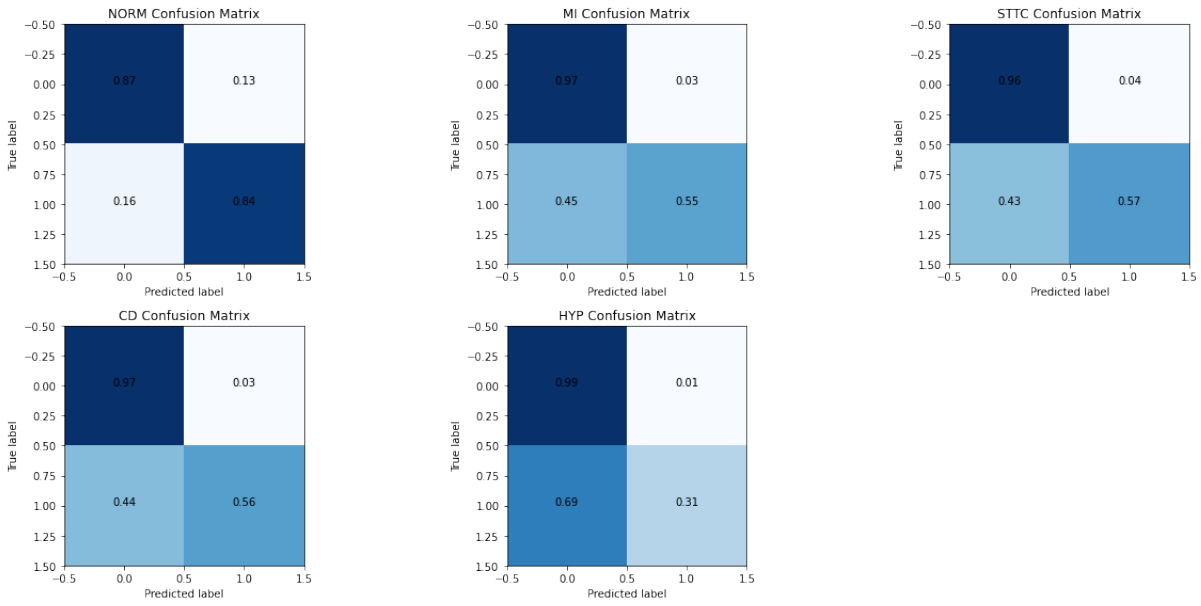

Figure 4.6: confusion matrix LSTM

the RNN model. This can be attributed to the LSTM's ability to capture long-term dependencies in the ECG signals, allowing it to better understand the underlying patterns and make more accurate predictions.

### 4.2.3 GRU model

The GRU model exhibits a true positive rate of 85% for the NORM class, indicating that it can accurately identify normal ECG patterns in a majority of cases, see figure4.7. While the true positive rates for the abnormal classes (MI, STTC, CD, and HYP) are higher than those observed for the RNN model, they are still lower compared to the LSTM model. This implies that the GRU model is better at identifying abnormal cases compared to the RNN model, but it does not perform as well as the LSTM model.

The false positive rates for the different classes range from 1% to 15%, with the highest rate observed for the NORM class. The true negative rates for the abnormal classes (MI, STTC, CD, and HYP) are higher than those for the RNN model, but they are still lower than the LSTM model. This suggests that the GRU model struggles to differentiate normal ECG patterns from abnormal cases to some extent, although it performs better than the RNN model.

Moreover, the false negative rates for the abnormal classes are lower than those for the RNN model, indicating that the GRU model can better identify instances of abnormalities within these classes. However, these rates are still higher than those observed for the LSTM model, suggesting that the LSTM model is more adept at detecting abnormalities in ECG signals.

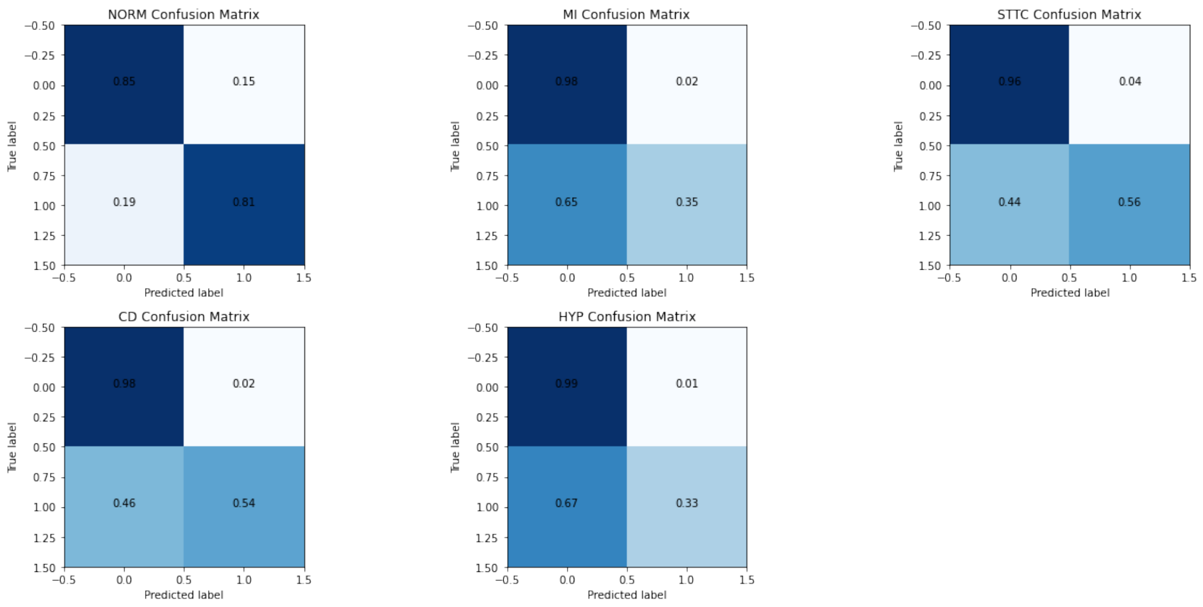

Figure 4.7: confusion matrix GRU

The true negative rates for the abnormal classes are higher than those for the RNN model but lower than those for the LSTM model. This demonstrates that the GRU model is better at distinguishing between different abnormal classes compared to the RNN model, but it still falls short of the LSTM model's performance.

In summary, the confusion matrix analysis for the GRU model reveals its improved performance in ECG classification compared to the RNN model, particularly in terms of identifying abnormal cases and differentiating between different classes. However, the LSTM model outperforms the GRU model in terms of overall classification performance, striking a better balance between true positive and true negative rates across all classes.

### 4.2.4 conclusion

The comparison of confusion matrices for the RNN, LSTM, and GRU models provides valuable insights into their respective performances in ECG classification tasks. The RNN model, while able to identify normal ECG patterns to some extent, struggles significantly with the classification of abnormal cases. This limitation emphasizes the need for more advanced models like LSTMs and GRUs in ECG classification tasks.

Both the LSTM and GRU models demonstrate improved performance compared to the RNN model, with the LSTM model exhibiting the best overall performance. The LSTM model strikes a better balance between true positive and true negative rates across all classes, demonstrating its robustness in accurately classifying ECG signals. The GRU model, although outperforming the RNN model, does not reach the same level of performance as the LSTM model.

These findings underscore the benefits of using LSTM and GRU models over traditional RNNs in ECG classification tasks. The LSTM model, in particular, demonstrates the most robust performance, making it a promising choice for the accurate detection of abnormalities in ECG signals. Further exploration into potential improvement techniques and optimizations for these models could lead to even better performance, enhancing their clinical utility in diagnosing cardiac conditions from ECG data.

# Chapter 5

# Discussion

## 5.1 The Influence of Architectural Design

The results of this study on ECG classification using the PTB-XL dataset show that the LSTM and GRU models outperform the RNN model in terms of accuracy, F1 score, precision, and recall. This difference in performance may be attributed to the unique architectural features of the LSTM and GRU models that are specifically designed to overcome the limitations of traditional RNNs. The basic RNN architecture is prone to the vanishing gradient problem, which hinders its ability to learn longer-term dependencies in the ECG data, whereas LSTM and GRU models are designed to address this issue [7][18].

## 5.2 Hyperparameters and their Effect on Performance

All three models used the same hyperparameters during training, which implies that the observed differences in performance are largely due to the choice of architecture. However, it is essential to consider that further optimization of hyperparameters could potentially improve the performance of each model. For instance, researchers could explore alternative learning rates, varying the number of hidden layers and units, or trying different optimization algorithms [4]. In-depth experimentation with these hyperparameters might reveal opportunities for enhancing each model's effectiveness in ECG classification tasks.

## 5.3 Insights from Confusion Matrices

The confusion matrices for the LSTM, RNN, and GRU models highlight the differences in their performances in ECG classification tasks. The LSTM model demonstrates the most robust performance, with higher true positive and true negative rates for all classes compared to the RNN and GRU models. The GRU model exhibits improved performance compared to the RNN model, particularly in terms of identifying abnormal cases and differentiating between different classes. However, it still falls short of the LSTM model's performance. The RNN model struggles with accurately classifying abnormal ECG patterns and distinguishing between various abnormal classes, which can be attributed to its architectural limitations, such as the vanishing gradient problem [3]. By analyzing the confusion matrices, it becomes evident that the LSTM and GRU models' unique architectural features enable them to overcome these limitations and achieve superior performance in ECG classification tasks.

## 5.4 Strategies for Enhancing Classification Performance

To further improve the classification performance, researchers can consider several strategies. First, refining the feature extraction and preprocessing techniques may enhance the models' ability to capture relevant information from ECG signals [25]. For example, exploring wavelet-based methods or advanced filtering techniques could yield better results. Second, incorporating domain knowledge into the model design can improve the models' interpretability and performance [48]. This can be achieved by using expert-designed features or integrating prior knowledge about the ECG signal structure into the models. Third, researchers can explore hybrid architectures, such as combining deep learning models with traditional machine learning algorithms like support vector machines (SVM) or random forests [32]. Hybrid architectures have been shown to achieve better performance in some ECG classification tasks compared to individual models alone.

Additionally, incorporating advanced techniques like data augmentation, transfer learning, ensemble methods, and attention mechanisms, as discussed earlier in the chapter, can further improve the models' performance. Researchers can also explore the use of other deep learning architectures, such as convolutional neural networks (CNNs) or transformer models, which have shown promise in ECG classification tasks [1, 37]. By considering these strategies and understanding the strengths and limitations of different models, researchers can continue to push the boundaries of ECG classification and ultimately improve the clinical utility of these algorithms.

## 5.5 Impact of Preprocessing and Feature Extraction

Another factor that might have influenced the performance of the models is the preprocessing and feature extraction of the ECG data. Different preprocessing techniques or feature extraction methods can impact the models' ability to learn and generalize from the data [25]. Exploring various approaches in this regard could lead to improved performance in ECG classification tasks. Researchers could also consider employing data augmentation techniques to increase the diversity and size of the dataset, which might help improve the models' performance and generalization capabilities.

## 5.6 Benefits of Transfer Learning and Ensemble Methods

Furthermore, researchers can explore the use of transfer learning to leverage pre-trained models and reduce the need for extensive training [46]. Transfer learning can help improve the performance of ECG classification tasks by utilizing knowledge learned from similar tasks or datasets. For example, a pre-trained model on a different ECG dataset can be fine-tuned to classify the specific categories of interest in the PTB-XL dataset. Another approach to enhance the performance of the models is to employ ensemble methods. Ensemble methods combine the predictions of multiple models to achieve better performance than any single model alone [30]. Researchers can experiment with various ensemble techniques such as bagging, boosting, or stacking to improve the classification performance for ECG signals [47].

## 5.7 Utilizing Attention Mechanisms

In addition, attention mechanisms can be incorporated into the models to allow them to focus on specific portions of the input data that are more relevant to the classification task [42]. Attention mechanisms have been shown to improve the performance of various tasks, including natural language processing and image classification, and could potentially enhance ECG classification models as well.

## 5.8 trade-offs: Performance, Complexity, and Computational Demands

It is also crucial to consider the trade-offs between model performance, complexity, and computational requirements. While LSTM and GRU models may demonstrate superior performance compared to the RNN model, they might also have increased complexity and computational demands. Assessing these trade-offs is essential in determining the most appropriate model for ECG classification tasks and can help inform the selection of the best method for ECG signal analysis.

## 5.9 conclusion

The choice of architecture, hyperparameters, preprocessing techniques, and various advanced methods such as data augmentation, transfer learning, ensemble methods, and attention mechanisms can significantly impact the performance of ECG classification models. By understanding the reasons behind the RNN model's limitations and experimenting with alternative approaches, researchers can continue to push the boundaries of ECG classification.

# Chapter 6

# Conclusions and future work

This research aimed at comparing the effectiveness of three different types of recurrent neural networks: basic RNN, LSTM, and GRU for the task of ECG signal classification into five categories: normal (NORM), myocardial infarction (MI), ST elevation (STTC), conduction disorder (CD), and hypertrophy (HYP). Data preprocessing and preparation was a crucial step, involving data loading, aggregation of diagnostic classes, data partitioning, and saving the preprocessed data for further analysis. This ensured the data was clean, organized, and ready for further analysis, thereby facilitating the successful training and evaluation of the RNN models. Each model architecture, RNN, LSTM, and GRU, was designed to handle the sequential nature of ECG data. The models were trained and validated, with their performance being evaluated using a separate validation dataset after each training epoch. Performance metrics such as accuracy, precision, recall, and F1 score were calculated to monitor the models' progress. The trained models were tested on a separate test dataset. The testing process was designed to evaluate the performance of the trained models on unseen data, providing a measure of their ability to generalize beyond the training data. The performances of the models were compared based on several metrics, including accuracy, precision, recall and F1-score, as well as their confusion matrices. The LSTM model achieved the highest accuracy of 87.89%, followed by the GRU model at 86.84%, and the basic RNN model at 76.08%. The LSTM model also outperformed the other two models in terms of precision, recall and F1-score, suggesting that it was the most effective model for the task.

A detailed analysis of the confusion matrices revealed more insights into the models' performance. The basic RNN model showed a high true positive rate for the normal class, indicating it was able to correctly identify most normal ECG patterns, but as seen in figure 4.5 we can't be sure it was accurate at all when it thought everything was NORM. It demonstrated a significant limitation in its inability to accurately classify any instances of the abnormal classes. This is a critical shortfall given the potential implications of misdiagnosing abnormal ECG signals in a clinical setting.

On the other hand, both the LSTM and GRU models demonstrated their ability to identify abnormal cases and differentiate between different classes. But even here, the LSTM model outshone the GRU model. It displayed a more balanced performance, achieving a better equilibrium between true positive and true negative rates across all classes. This balance is crucial because it indicates the model's ability to accurately identify both the presence and absence of a condition, a crucial factor in medical diagnoses.

In summary, our findings indicate the superiority of the LSTM model over both the basic RNN and GRU models for the task of ECG signal classification. Its robust performance across multiple metrics and its balanced classification rates underscore its potential utility in the detection and diagnosis of cardiac conditions.

we recognize that further work is needed. Future research should explore potential improvement techniques and optimizations for these models. This could include hyperparameter tuning, which involves adjusting the parameters used in the model's learning process to optimize performance. Ensemble methods, which combine the predictions of multiple models to improve accuracy, could also be investigated. Additionally, integrating these models with other types of neural networks may yield even better results. In addition, incorporating these models with other types of neural networks might yield even better results. For example, Convolutional Neural Networks (CNNs) have shown promise in time series analysis and could potentially be used alongside or in combination with LSTM models to improve classification accuracy. Moreover, though our models' performance was evaluated using several metrics, there may be other important aspects to consider. For example, in a real-world clinical setting, the cost of misclassification (especially false negatives) can be extremely high. As such, future work could include an investigation into cost-sensitive learning or other methods to minimize these potentially harmful errors.

In conclusion, the field of ECG signal classification presents numerous opportunities for the application of machine learning, and specifically RNNs. Through our analysis, we found that LSTM models provide the most promising results among the three types of RNNs evaluated. However, this should not diminish the potential of other models, such as GRUs, especially when considering future optimizations and improvements. As we move forward, the adoption of these advanced machine learning models holds the potential to significantly enhance the speed and accuracy of cardiac disorder diagnosis, ultimately leading to better patient outcomes.

# Bibliography

[1] U Rajendra Acharya, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan and Hojjat Adeli. 'Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals'. In: *Computers in biology and medicine* 100 (2018), pp. 270–278.

[2] Zachi I Attia, Peter A Noseworthy, Francisco Lopez-Jimenez, Samuel J Asirvatham, Abhishek J Deshmukh, Bernard J Gersh, Rickey E Carter, Xiaoxi Yao, Alejandro A Rabinstein, Brad J Erickson et al. 'An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction'. In: *The Lancet* 394.10201 (2019), pp. 861–867.

[3] Yoshua Bengio, Patrice Simard and Paolo Frasconi. 'Learning long-term dependencies with gradient descent is difficult'. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[4] James Bergstra and Yoshua Bengio. 'Random search for hyper-parameter optimization.' In: *Journal of machine learning research* 13.2 (2012).

[5] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN: 9780387310732. URL: https://books.google.no/books?id=qWPwnQEACAAJ.

[6] Davide Chicco and Giuseppe Jurman. 'The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation'. In: *BMC genomics* 21 (2020), pp. 1–13.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio. 'Learning phrase representations using RNN encoder-decoder for statistical machine translation'. In: *arXiv preprint arXiv:1406.1078* (2014).

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho and Yoshua Bengio. 'Empirical evaluation of gated recurrent neural networks on sequence modeling'. In: *arXiv preprint arXiv:1412.3555* (2014).

[9] Jeffrey L Elman. ‘Finding structure in time’. In: *Cognitive science* 14.2 (1990), pp. 179–211.

[10] K Anders Ericsson. ‘Deliberate practice and the acquisition and maintenance of expert performance in medicine and related domains’. In: *Academic medicine* 79.10 (2004), S70–S81.

[11] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng and H Eugene Stanley. ‘PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals’. In: *circulation* 101.23 (2000), e215–e220.

[12] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep learning*. MIT press, 2016.

[13] Michael Green, Jonas Björk, Jakob Forberg, Ulf Ekelund, Lars Edenbrandt and Mattias Ohlsson. ‘Comparison between neural networks and multiple logistic regression to predict acute coronary syndrome in the emergency room’. In: *Artificial intelligence in medicine* 38.3 (2006), pp. 305–318.

[14] Isabelle Guyon and André Elisseeff. ‘An introduction to variable and feature selection’. In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[15] Amir Halkin, Mandeep Singh, Eugenia Nikolsky, Cindy L Grines, James E Tcheng, Eulogio Garcia, David A Cox, Mark Turco, Thomas D Stuckey, Yingo Na et al. ‘Prediction of mortality after primary percutaneous coronary intervention for acute myocardial infarction: the CADILLAC risk score’. In: *Journal of the American College of Cardiology* 45.9 (2005), pp. 1397–1405.

[16] Elias B Hanna and David Luke Glancy. ‘ST-segment depression and T-wave inversion: classification, differential diagnosis, and caveats’. In: *Cleveland Clinic journal of medicine* 78.6 (2011), p. 404.

[17] Awni Y Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia and Andrew Y Ng. ‘Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network’. In: *Nature medicine* 25.1 (2019), pp. 65–69.

[18] Sepp Hochreiter and Jürgen Schmidhuber. ‘Long short-term memory’. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[19] Mohammad Kachuee, Shayan Fazeli and Majid Sarrafzadeh. ‘Ecg heartbeat classification: A deep transferable representation’. In: *2018 IEEE international conference on healthcare informatics (ICHI)*. IEEE. 2018, pp. 443–444.

[20] John D Kelleher, Brian Mac Namee and Aoife D'Arcy. 'Fundamentals of machine learning for predictive data analytics: algorithms'. In: *Worked examples, and case studies* (2015).

[21] Paul Kligfield, Leonard S Gettes, James J Bailey, Rory Childers, Barbara J Deal, E William Hancock, Gerard Van Herpen, Jan A Kors, Peter Macfarlane, David M Mirvis et al. 'Recommendations for the standardization and interpretation of the electrocardiogram: part I: the electrocardiogram and its technology: a scientific statement from the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society endorsed by the International Society for Computerized Electrocardiology'. In: *Circulation* 115.10 (2007), pp. 1306–1324.

[22] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. 'Deep learning'. In: *nature* 521.7553 (2015), pp. 436–444.

[23] Cuiwei Li, Chongxun Zheng and Changfeng Tai. 'Detection of ECG characteristic points using wavelet transforms'. In: *IEEE Transactions on biomedical Engineering* 42.1 (1995), pp. 21–28.

[24] Zachary C Lipton, David C Kale, Charles Elkan and Randall Wetzel. 'Learning to diagnose with LSTM recurrent neural networks'. In: *arXiv preprint arXiv:1511.03677* (2015).

[25] Eduardo José da S Luz, William Robson Schwartz, Guillermo Cámara-Chávez and David Menotti. 'ECG-based heartbeat classification for arrhythmia detection: A survey'. In: *Computer methods and programs in biomedicine* 127 (2016), pp. 144–164.

[26] Tom M Mitchell. 'Does machine learning really work?' In: *AI magazine* 18.3 (1997), pp. 11–11.

[27] George B Moody, Roger G Mark and Ary L Goldberger. 'PhysioNet: a web-based resource for the study of physiologic signals'. In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 70–75.

[28] Jiapu Pan and Willis J Tompkins. 'A real-time QRS detection algorithm'. In: *IEEE transactions on biomedical engineering* 3 (1985), pp. 230–236.

[29] Razvan Pascanu, Tomas Mikolov and Yoshua Bengio. 'On the difficulty of training recurrent neural networks'. In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318.

[30] Robi Polikar. 'Ensemble based systems in decision making'. In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[31] Piotr Ponikowski, Adriaan A Voors, Stefan D Anker, Héctor Bueno, John GF Cleland, Andrew JS Coats, Volkmar Falk, José Ramón González-Juanatey, Veli-Pekka Harjola, Ewa A Jankowska et al. '2016 ESC Guidelines for the diagnosis and treatment of acute and chronic heart failure'. In: *Kardiologia Polska (Polish Heart Journal)* 74.10 (2016), pp. 1037–1147.

[32] Pranav Rajpurkar, Awni Y Hannun, Masoumeh Haghpanahi, Codie Bourn and Andrew Y Ng. 'Cardiologist-level arrhythmia detection with convolutional neural networks'. In: *arXiv preprint arXiv:1707.01836* (2017).

[33] Saeed Saadatnejad, Mohammadhosein Oveisi and Matin Hashemi. 'LSTM-based ECG classification for continuous monitoring on personal wearable devices'. In: *IEEE journal of biomedical and health informatics* 24.2 (2019), pp. 515–523.

[34] Tara N Sainath, Oriol Vinyals, Andrew Senior and Haşim Sak. 'Convolutional, long short-term memory, fully connected deep neural networks'. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. Ieee. 2015, pp. 4580–4584.

[35] Tommaso Sanna, Hans-Christoph Diener, Rod S Passman, Vincenzo Di Lazzaro, Richard A Bernstein, Carlos A Morillo, Marilyn Mollman Rymer, Vincent Thijs, Tyson Rogers, Frank Beckers et al. 'Cryptogenic stroke and underlying atrial fibrillation'. In: *New England Journal of Medicine* 370.26 (2014), pp. 2478–2486.

[36] Jürg Schläpfer and Hein J Wellens. 'Computer-interpreted electrocardiograms: benefits and limitations'. In: *Journal of the American College of Cardiology* 70.9 (2017), pp. 1183–1192.

[37] Supreeth Prajwal Shashikumar, Amit J Shah, Qiao Li, Gari D Clifford and Shamim Nemati. 'A deep learning approach to monitoring and detecting atrial fibrillation using wearable technology'. In: *2017 IEEE EMBS international conference on biomedical & health informatics (BHI)*. IEEE. 2017, pp. 141–144.

[38] Shraddha Singh, Saroj Kumar Pandey, Urja Pawar and Rekh Ram Janghel. 'Classification of ECG arrhythmia using recurrent neural networks'. In: *Procedia computer science* 132 (2018), pp. 1290–1297.

[39] Marina Sokolova and Guy Lapalme. 'A systematic analysis of performance measures for classification tasks'. In: *Information processing & management* 45.4 (2009), pp. 427–437.

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. 'Dropout: a simple way to prevent neural networks from overfitting'. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[41] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017).

[43] Galen S Wagner, Peter Macfarlane, Hein Wellens, Mark Josephson, Anton Gorgels, David M Mirvis, Olle Pahlm, Borys Surawicz, Paul Kligfield, Rory Childers et al. 'AHA/ACCF/HRS recommendations for the standardization and interpretation of the electrocardiogram: part VI: acute ischemia/infarction: a scientific statement from the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society: endorsed by the International Society for Computerized Electrocardiology'. In: *Circulation* 119.10 (2009), e262–e270.

[44] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek and Tobias Schaeffter. 'PTB-XL, a large publicly available electrocardiography dataset'. In: *Scientific data* 7.1 (2020), p. 154.

[45] Qihang Yao, Ruxin Wang, Xiaomao Fan, Jikui Liu and Ye Li. 'Multi-class arrhythmia detection from 12-lead varied-length ECG using attention-based time-incremental convolutional neural network'. In: *Information Fusion* 53 (2020), pp. 174–182.

[46] Jason Yosinski, Jeff Clune, Yoshua Bengio and Hod Lipson. 'How transferable are features in deep neural networks?' In: *Advances in neural information processing systems* 27 (2014).

[47] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[48] Zhancheng Zhang, Jun Dong, Xiaoqing Luo, Kup-Sze Choi and Xiaojun Wu. 'Heartbeat classification using disease-specific feature selection'. In: *Computers in biology and medicine* 46 (2014), pp. 79–89.

# Appendix A

# Github Repository

All code used in thesis: https://github.com/sandersaether/Master