Automated Wearable ECG Data Editing with DNN

A Major Qualifying Project (MQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements
for the Degrees of Bachelor of Science in

Electrical and Computer Engineering
Computer Science

By:

Vincenza Burdulis, Jacob Nguyen, Daniel Sardak

Project Advisor:

Bashima Islam

Date: April 2024

Abstract

Electrocardiograms (ECG) are frequently used to check the heart's health. Using electrodes, the electrical activity of the heart is able to be recorded. While this is usually performed by a professional, there are also wearable ECG devices. There's different forms of the device, but each allows for an easy way to analyze the heart. A common problem with wearable electrocardiograms is that they aren't always accurate due to various reasons. To solve this issue a technician is able to manually edit the data to extract any noise or artifacts. Manually editing is a time consuming technique due to the large volume of data. Automating this process would significantly improve the efficiency and accuracy. This project develops a deep learning model to automate the data editing process for wearables. Our project, had tested out multiple machine learning models to see which had the highest performance. Through testing, we found that the Bidirectional Recurrent Neural Networks (Bidirectional RNN) and the Long Short Term Memory (LSTM) models had worked best with our data as they were able to detect over half the anomalies.

Executive Summary

Electrocardiograms (ECGs) are used to detect the health of a patient by analyzing the different waves shown on the graph. These different waves can signal how an individual's body is performing based on your heart rate and rhythm. Data that is often looked at from ECGs are interbeat intervals, or IBI data. This IBI data can be used to measure an individual's autonomic balance, blood pressure, gas exchange, gut, heart, and vascular tone. However, this data collection process is imperfect, and can collect incorrect data (anomalies) resulting from the Biopac either missing datapoints, resulting in some IBI values being abnormally high, and from noise introduced through usere movement throughout the collection process. This means that before analyzing IBI data, it must first be edited to remove these anomalies. Currently, the IBI data collected from ECGs is manually edited for anomalies. To do so, the editor must identify the artifact based on a given criteria and then correct by integer division. Our project looks to automate this process through the use of deep learning. We chose to use examine a deep learning approach because deep learning models have been shown to perform very well in general classification tasks, and there have been many advances made in deep learning-based signal processing approaches. We explore 4 different DNN architectures to identify anomalies in IBI data.

We got our data from Dr. Nancy McElwain from the University of Illinois Urbana-Champaign. While the dataset we were provided included data from infants of several different ages, to ensure consistency in results we focus primarily on data from 3 month old children. This gave us 396 IBI files to evaluate. To make the task of training a neural network simpler, we split each user's IBI sequence into chunks of 20 datapoints for easier evaluation. However, we did make sure to keep each user's respective data either in our training set or testing set as splitting each user up amongst the two sets led to poor model performance.

The two models that performed the best for us were the Bidirectional Vanilla Recurrent Neural Netowrk (RNN) and Long Short Term Memory (LSTM) architectures. With these models, we implemented a correction algorithm to edit the anomalies in the given IBI data. This algorithm includes a simple signal processing method guided by core IBI characteristics. From this, we were able to determine that our correction algorithm

Model	RNN	LSTM
F1 Score	0.75	0.80
Anomoly Accuracy	78%	83%
Clean Data Accuracy	94%	90%
Kappa	0.72	0.70

Table 1: Results for best performing models

When testing out the different machine learning models, we had focused on the four different criteria shown above. The models that had achieved a higher score, were the ones we focused on to try and correct the data.

Acknowledgements

We would like to thank our advisor Professor Bashima Islam and PhD Student Mohammad Nur Hossain Khan. We would also like to thank UIUC School of Medicine for providing us with the data and software necessary for our project. Additionally we would like to thank Worcester Polytechnic Institute for providing us with the opportunity to complete this project.

Contents

List of Tables

List of Figures

1 Introduction

Electrocardiograms (ECG's) record the electrical signals in the heart which shows the rhythm of the heart and how fast it beats. One thing that is often looked at is the interbeat interval (IBI) data as it's a significant health indicator. Health providers look into several characteristics of the recorded ECG data. It can give important information on the health of a patient's heart, such as finding if there are any preexisting conditions with a person's heart. They are most commonly used to diagnose conditions with the heart's electrical system, cardiac arrest, heart attacks, and other heart related conditions. ECG data is used to see how a person's heart functions. The test is administered by attaching electrodes to a patient and sending electrical signals throughout their body.

Though medical grade ECG devices at hospitals are accurate, they are costly and bulky. These wearable ECG devices are gaining popularity to bring such important and vital monitoring outside the four walls of the hospital. These wearable ECG monitors however often give erroneous data that needs to be edited. This is due to the sensitivity and specificity of the devices as these devices detect a large number of false positives[24].

Oftentimes this data needs to be edited to remove any artifacts to allow for an accurate reading. To edit interbeat interval (IBI) data, experts need to review the data manually, and find and correct any artifacts. Artifacts are introduced from both the electrocardiography (ECG) data collection phase, and from the conversion from ECG to IBI data. The amount and severity of these artifacts is further increased when one collects data from portable devices, which makes data more prone to data collection errors and motion artifacts.

Correcting these artifacts is a very tedious and time consuming process, as the accurate processing of artifact detection and correction depends more on the intuition and experience of the editor than any strict algorithmic approach. For instance, a 5 minute period of IBI data can take upwards of 20-30 minutes to properly analyze and correct.

In order to simplify and automate this process, our project seeks to develop a deep learning model to automatically detect and correct any artifacts in IBI data. Our model is using data from infants aged from 3-12 months, provided by UIUC, the University of Illinois Urbana-Champaign.

We developed and tested multiple models to help with this process. The models that had the best performance were a Bidirectional RNN (Recurrent Neural Network) and a LSTM (Long Short Term Memory). The Bidirectional RNN was able to correctly predict anomalies at a 78% accuracy, and the LSTM was able to predict at a 83% accuracy. Using these models, we implemented and modified a correction algorithm provided to us by Professor Bashima Islam to automate the IBI editing process.

2 Literature Review and Background Study

2.1 Electrocardiogram Data Extracted Information

Electrocardiogram (ECG): An electrocardiogram, also known as an ECG or EKG, is used to record electrical signals from an individual's heart in order to determine its condition or status. This data is used to find any existing problems or diseases that are affecting a person's heart. ECGs are most commonly used to detect cardiovascular disease, as they can detect the rate a person's heart is beating, the rhythm of a person's heart, and the strength and timing of the electrical pulses that pass through the heart. On an ECG, there are 6 different waves that are represented on the graph as shown in figure 1: the P wave, Q wave, R wave, S wave, T wave, and U wave. The P wave is the first upward waveform, the Q wave is a downward deflection, the R wave is the upward peak or spike, and the S wave is a downward wave. The most common intervals that are measured on an ECG are the PR, QRS, QT, and RR intervals. These intervals are used to provide information on an individual's heart activity, rate, and rhythm. Each interval indicates different stages: the P wave indicates atrial depolarization, the QRS complex indicates ventricular depolarization, the T wave indicates ventricular repolarization, and the QT interval represents the time taken between ventricular depolarization and ventricular repolarization [1].

ECG devices have three or more electrodes attached to an individual's chest, arms, and legs. The electrodes measure electrical activity in the heart through leads that are attached to the ECG machine [1]. There are 2 main types of ECG monitors for long term use: Holter monitors, and event monitors. A Holter monitor is a portable ECG that can record the heart's activity at periods 24 hours or longer. Event monitors are also used to record similar information to ECGs, but are used for longer periods of time. These devices often transmit the collected data to a patient's healthcare provider or doctor. They are used when a patient is suspected to have an abnormal heart rate based on their medical history [5]. In today's technological world, ECGs can be taken with wearables such as smart watches, which increases the accessibility of these tests and data. However, portable and accurate ECG devices are quite costly. They often are less expensive, but a little noisy ECG devices are often used.

Interbeat Interval (IBI): Interbeat Interval, known as IBI, is the time between beats of an individual's heart and is used to measure one's Heart Rate Variability (HRV). These measurements are taken from

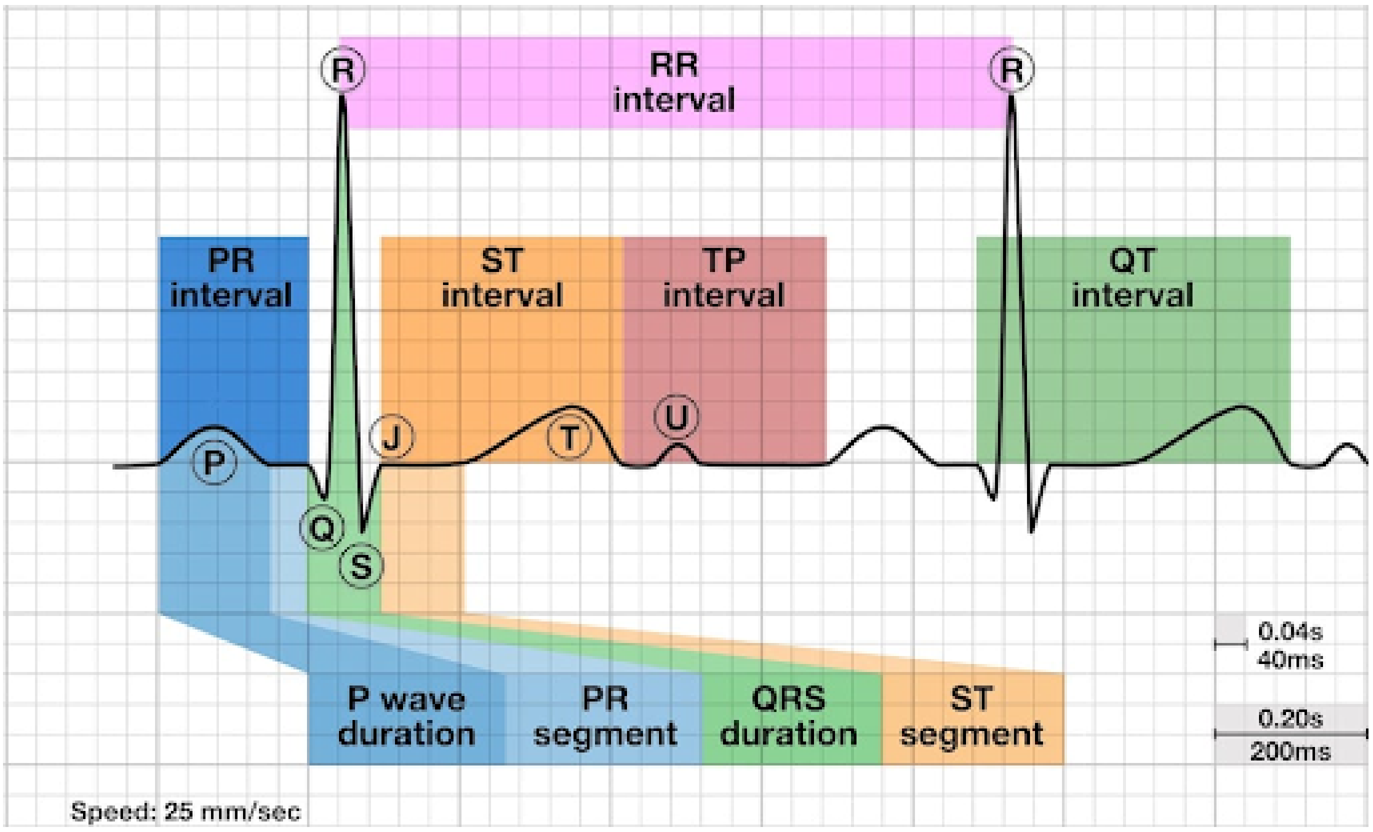

Figure 1: Electrocardiogram Labeled, LITFL [2]

an electrocardiogram and is the time between R peaks, or the RR interval. Heart Rate Variability is the oscillation in time between consecutive heartbeats. It is an indication for an individual's autonomic balance, blood pressure, gas exchange, gut, heart, and vascular tone. There are influences that can affect HRV, which are parasympathetic influences, sympathetic influences, blood pressure, baroreceptors, thermodynamics, and basic metabolic processors [3]. A healthy heart does not follow a consistent and rigid pattern, as it is healthy for oscillations in heart beats to occur. This indicates that the heart is able to adapt to different situations and environments, depending on what is needed. Therefore, having a variation of time intervals between a person's heart beats is an indication that the individual has a balanced autonomic system. Having a balanced autonomic system reflects that the body can handle and adapt to stressors, internally and externally. It is important to measure HRV because it is an indicator of a person's physical health, cognitive, and emotional state. It identifies how an individual reacts to or recovers from stimuli, and can also predict social and emotional development. Normal high frequency HRV values range from 0.3 to 1.3 Hz in infants, with a range of 0.06 - 0.10 Hz for all ages. Normal interbeat intervals range anywhere from 300 ms to 500 ms in infants.

2.2 Noisy IBI Data Correction

Noisy data of low-cost wearable ECG devices provided inaccurate IBI data, which needs to be corrected. In order to ensure clean IBI data, one must first ensure that their ECG data is clean, and all R peaks were successfully detected. Since IBI data is directly derived from ECG data, having good starting data is key to having clean IBI data with little editing required.

ECG data is edited by analyzing it and identifying every R peak contained within the data. While there are several good automatic detection algorithms available (such as thresholds across the time period or across smaller subsections of the data), no approach can be perfect, and the enormous impact missing even one R peak can have on IBI data and its resulting HRV requires an editor to manually check the R Peak detection. Having working and successful R peak detection is important when creating an IBI editing model, so that it limits the amount of edits that are needed in the data. There are countless such programs to manually review ECG data, such as CardioEdit. Once in the program, an editor must scan the data to ensure that all peaks were properly detected, and that no additional points were marked as peaks. The primary tools an editor has to correct this data is removal and placement; an editor can remove a detected R peak, excluding it from IBI calculation, or can mark a new one, adding a point to calculate.

Once an editor has marked all of the ECG's R peaks, they can then convert the data to IBI data. Because of interbeat interval data's direct relation with HRV, an editor should aim to make the least amount of edits possible to ensure the integrity and maintain the validity of any resulting HRV calculations. A maximum number of edits a user should make is approximately 5-10% of the original IBI dataset. Editing more can result in the data being overedited and the calculated HRV to be artificially low.

In order to detect any artifacts, an editor must analyze IBI data, and look for any abnormal points. In general, as many of these artifacts result from not a misdetected R peak, artifacts tend to take the shape of one point shooting up with a significantly higher corresponding value than its surrounding points. In order to correct these artifacts, the editor has 3 primary tools: averaging, dividing, and deleting.

When the IBI data in a segment seems to follow a certain trend, but have a single point disrupting that trend, averaging can be a useful tool to make the artifact fit the data. The editor would take a couple points on each side of the artifact and average them, setting the artifact's value to that average. While editing, it is important to note the difference between artifacts and arrhythmias. Although a data point may look like an artifact, looking at the surrounding data to make sure the point is not a true outlier is important. Spikes or drops in data points can be a result of the patient's environment, as factors such as stress, anxiety, or physical activity can alter the interbeat interval.

The other primary tool an editor has is dividing by N. When a certain interval appears to be significantly higher than everything else (for instance, if the reader missed several R peaks leading to a single interval being significantly greater than its surrounding values), an editor may divide the artifact by the number of missed peaks to make the data fit in well with everything else. This may result in consecutive points with equal values, but this edited data is better than having the artifact present.

The final tool editors can use is deletion. However, because of the severity and impact on HRV that deletion can have, this should be a last measure only used when the other options do not work, or there is a whole chunk of unusable data in a time segment. When data is unusable, or is so full of artifacts that the original trends of the data can not be determined without editing nearly every point, an editor can choose to cut out a segment of data and to exclude it from evaluation.

2.3 Deep Learning

Deep learning is a branch of machine learning, which is a neural network with 3 or more layers. These networks simulate the behavior of the human brain in order to learn from a large data set. Artificial intelligence (AI) allows for machines and computers to process, analyze and make decisions like a human would. It can be seen in different forms differing from self-driving cars to using Siri on your iPhone [16]. AI is becoming more popular in healthcare as it is able to help analyze specific data and foresee any challenges. More specially, deep learning algorithms can be applied to certain medical problems [16]. Deep learning is a method of artificial intelligence, but more specifically it is a machine learning technique. Computers are able to solve problems involving speech, text and sound [4]. This method is generally made up of artificial neural networks (ANN) that use multiple processing layers. These patterns allow for patterns and structure in large data sets to be discovered. Every layer is able to build on itself based on the previous layer allowing the higher layers to have more abstract concepts [15]. The purpose of these layers is that each layer is created by data, not humans [14]. An example of this would be an image having and using neural networks to denoise the image. Figure 2 shows this example.

Within deep learning, there are different types of algorithms that serve different purposes. Two that are often used for ECGs are convolutional neural networks (CNN) and Recurrent Neural Networks (RNN). Convolutional neural networks process data that comes from multiple arrays and deals with properties from natural signals. They're common for image processing and reduce the number of parameters to be learned [12]. The convolutional layer is the first of the network and this is where the computation occurs. An input is given and it'll go through different layers/filters for the output to be produced. Depending on how

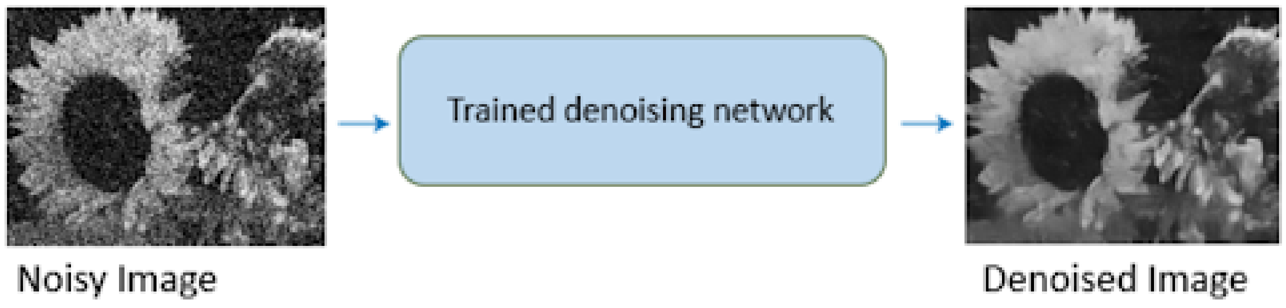

Figure 2: Example of denoising imagine using deep learning [13]

many layers or filters will determine the outcome of the output [11]. Pooling also can occur as it allows for translation invariance on the learned features. This can be applied as many times as needed which can result in connected layers [12]. Other forms of CNN are Tiled Convolutional Neural Networks and Fully Convolutional Networks (FCN). FCN allows the input and output layer to have the same dimensions while tiled allows for groups to be divided into sub grounds which have separate weights [12].

Recurrent Neural Networks (RNN) uses sequential or time series data. RNN trains the data and has the output be based on elements that occur earlier in the sequence This network is able to share the same weight parameter throughout unlike feedforward ones [9]. Within the network, the backpropagation algorithm can be used to help train the recurrent neural networks by constraining some of the connections that hold the same weight [12]. There is also a backpropagation through time (BPTT) algorithm that helps the network determine gradients while still using the same technique as the original algorithm [9]. Although, using these algorithms issues can occur such as the gradient being too small or too large which is referred to as vanishing gradient. A solution to this issue can be to use a Long Short Term Memory (LSTM) architect [12]. With this, the hidden layer has cells with three different gates, input, output and forget. The different gate allows the information to be controlled so the output can be predicted [9]. The neural machine translation system generally has an encoder and decoder structure. This is often used in sequence to sequence problems as the input and output are varying lengths.

2.4 How to Process IBI Data using DNN

In order to train a deep neural network on IBI data, we require a dataset of clean and flawed IBI data. Since our project will focus on taking flawed data and correcting any artifacts, we will use flawed IBI data with artifacts as input, and our output and ground truth will be a clean version of the IBI data without any artifacts. In order to obtain clean data, we will use several public datasets focused on IBI, such as the MIT-BIH Arrhythmia Database (DB)

One way to fix created noisy ECG data is by adding weights and values to good data to simulate the noise and artifacts created in the data collection process. One such formula is $ECG_{noisy} = ECG_{clean} + a_1\text{EM} + a_2\text{MA} + a_3\text{BW}$, where a_1, a_2, a_3 represent weights,EM represents electrode motion noise, MA represents motion artifact noise, and BW represents baseline wandering noise (2107.00693.pdf (arxiv.org)). As our work focuses on BioPacks, which suffer from motion artifact noise and electrode motion noise, we will likely weight those numbers higher. In addition to generating our own clean and raw data, the MIT-BIH database also has some raw data in addition to clean data we can use to generate more information.

Since IBI data can be classified as time series data (it represents the time difference between heartbeats over time), we researched general deep learning techniques applied to time series data to see if any of those algorithms can be applied to IBI data. There are currently several different approaches to analyzing general time series data with deep learning. These include converting the data to an image and using computer vision algorithms,using Fully Convolutional Networks, and using Recurrent Neural networks.

Since a lot of time data has the same number of inputs as outputs (every heartbeat has one interval between itself and the next heartbeat), some experts suggest a Fully Convolutional Network (FCN) as opposed to the traditional CNN [20]. FCNs can fit time series data better than CNNs since they feature a deconvolution stage, which allows the model to match input and output parameter size.

Another common deep learning technique used to model time series is RNNs [23]. Since each layer of an RNN builds on the last, these networks can be good for time series data (which by definition is sequential), allowing for a better fit than CNNs or other model types. However, RNN's dependency on previous instances of data makes it more prone to issues such as gradient disappearance and explosion (NGCU: A New RNN Model for Time-Series Data Prediction - ScienceDirect). While these issues are not insignificant, the previously mentioned advantages of RNNs still makes them a good choice for different types of time series data.

One proposed architecture that was shown to forecast well on multiple different datasets is the N-BEATS [18]. This deep learning architecture is built on a multi-layer FC network, and is organized into stacks, with each layer outputting partial forecasts and backcasts, and the final output of an element being a summation of all the partial predictions. This architecture's complexity helps the model generalize to new time series data better, with many of its optimal architecture designs and hyperparameters being able to apply to many different time series at once.

Another alternative model type to forecast and classify time series data is transformers [19]. Transformers, and their ability to "capture long-range dependencies and interactions is especially attractive for time series modeling" has led to a significant burst in their popularity, and a growing set of "basic" transformer modifications and expanded architectures to better suit time series data. The general structure of a transformer is split into two stages: encoding and decoding. The encoding stage transforms the input data by positionally encoding each value so the model gains information about the order of the sequence entries. After initially transforming the data, the model then uses multi-head self-attention to convert the data to a usable form. The decoder has a similar structure, where it first embeds outputs, runs mult-head self-attention on that data, and then combines those resulting weights with output from the encoder to generate output data. Some of the benefits of the transformer are its increased parallelizability for faster and more efficient training as opposed to traditional RNNs, and its versatility in being able to model a lot of different data types well.

2.5 Existing Works

Electrocardiograms (ECGs) are prone to being affected by noise and artifacts when being worn. This creates error signals in the data, but can be manually fixed by an ECG technician. To help with the process, multiple softwares have been created to deal with these inaccuracies. ARTiiFACT is a software that processes both IBI and electrocardiogram data. This software offers both automated and manual artifact detection and correction. It also includes time and frequency based HRV analysis tools. IBI needs to be extracted for the HRV to be analyzed to which the R-peak is extracted from digitized ECG data. Within the program it provides tools that allow the user to have a complete range of data processing. For R-Peak detection ARTiiFACT provides the ability to low pass or high pass filter raw ECG data at a manually adjusted frequency. With the detection, you can choose between a global or local threshold criteria. This feature allows you to detect drifts and exceeding amplitude in the data. When it comes to identifying false IBI data, an algorithm is implemented. The algorithm derives artifact detection from the different distributions and then it will assess and remove. From the main window ARTiiFACT, it has four subcomponents that cover the full steps of processing raw ECG data. It was developed with MATLAB and the software is free with request.

In the European Signal Processing Conference, a paper was presented discussing two different deep learning models that were used to denoise ECG data. The models that were used were Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM). Both of these models were as well constructed in MATLAB. The convolutional neural networks had been used before to denoise ECG's, but not reconstruct.

With the model that was used, it had six two dimensional layers with each layer having thirty six kernels. Each layer was made to reduce the dimensionality of the input ECG signals. With this model it took about ten minutes for the root square mean to be calculated and roughly fifty eight to reach an average RMS. The second model that was tested was LSTM. This model consisted of 140 hidden nodes per layer with the first layer having a similar dimension as the ECG sequence. Compared to CNN, the LSTM's performance was a bit less impressive. The convolutional neural network was able to obtain a RMS lower than the long short term memory. Along with that it took less time to compute and required a smaller amount of training samples.

Another model we looked at was an RNN Time-Series Anomaly Detection found on GitHub. It was implemented in PyTorch, and has a two-stage strategy made up of time-series prediction, and calculating an anomaly score. The model had 4 different elements that made up the code: 1_train_predictor, 2_anomoly_detection, anomalyDetector, and Preprocess_data. The first element contained the RNN model for the prediction. It contained important aspects of the training such as batch generation, loss calculation, backpropagation, optimization, and logging. The mean and covariance of the data is calculated after training is run, which are pieces of data that are used in the anomaly detection. The 2nd element of the model calculated the anomaly score, which is calculated based on the dataset and previously calculated mean and covariance. The predictor is trained using support vector regression (SVR), and precision, recall, and F1-scores were used to evaluate the score. The 3rd element calculated the mean and covariance of the prediction errors of the dataset. To do so, the detector iterates through the dataset using an RNN model, collects and organizes the predictions to calculate errors, and calculates the mean and covariance of the dataset.

The final piece of the model preprocesses the dataset used. The sequence data is normalized using a min-max scaling, and uses standardized mean and standard deviation. One important function in this element is PickleDataLoad. There are 4 steps that occur in this function, which are initialization, augmentation, preprocessing, and batchify. In the first step, the training and test data is loaded and preprocessed. During the augmentation stage, the data is augmented with Gaussian noise and is added to the original sequence. After this, in preprocessing, the mean and standard deviation is calculated if it is in training mode, and the data is augmented and standardized using the computed mean and standard deviation. Finally, the data is divided into batches for training and test, and this batched data is converted to the selected device.

3 Methodology

Our research shows us the general process of IBI editing, and what tools editors have at their disposal to clean up files, and fix issues with IBI sequences. In order to simplify the tasks for our models, we split this task into stages: classification, and correction. The classification stage will go through and detect any of the data points that are outliers. Then the correction stage will fix these anomalies by dividing the point so it will match the surrounding data.

We primarily focused on the detection stage of the process. As IBI files can be of varying lengths, we approached the problem by separating full sequences into shorter, fixed length sequences.

In order to correct our classified anomalies, we employed a signal processing based IBI data correction algorithm. In the following sections, we go into more detail about the algorithm used.

3.1 Pre-Processing

To help us understand what issues we may run into while making our models, we make an initial set of models to see how they would perform on the task. After seeing how high initial accuracy is despite the model only predicting 1 class, we realized that our data was far too unbalanced to judge with accuracy, leading to using many different metrics for our task and settling with F1 score as a primary performance metric for each model. To simplify the predictions required, we adjusted our initial models to only classify sequences as containing anomalies or not containing anomalies. Additionally, we adjusted our train-test split to group by files and not sequences to ensure the model would be able to either test or learn from a full stream and that we would not have part of a file in the test bucket, and another part of the same file in the training dataset.

We also experienced better results with sub-sampling and oversampling. In order to generate more sequences containing anomalies, we created a moving window algorithm that would scan through each anomaly instance, and create new sequences by sliding across whichever sequence the anomaly was in. We experimented with creating different amounts of data points, and found that models would have higher F1-scores with 2 new sequences for each anomaly.

The final adjustment to our data we made was changing the input sequence size. Testing showed that models would generally perform with sequences length 10 or 25 than 5 or anything greater. We believe this is caused by shorter sequences not having enough information for the model to learn off of, and longer sequences having too much clean data to properly detect anomalies.

3.2 Proposed Model

We developed three models for correction - the Bidirectional Vanilla RNN, the LSTM and Bidirectional LSTM. The following section goes through each of the models explaining each model's architecture, our process for tuning each model.

3.2.1 Bidirectional Vanilla RNN

As our data is time series data, and RNNs can leverage the sequential nature of our data to create better predictions, we created a model with multiple RNN layers. A Bidirectional Vanilla RNN is designed to process sequences in both forward and backward directions allowing the model to look at all stages of the sequence, which is especially useful since anomalies are based on all points of the sequence and not just points before or after them in each sequence.

In order to avoid potential over fitting, we worked to not over complicate the model, and focused on making the best model with simplest possible architecture.

3.2.2 Long Short Term Memory

One of the models that we tested with our data was a Long Term Short Memory (LSTM) Model. It is a type of RNN that can address the gradient problem that often occurs in RNNs. This gradient makes it difficult to capture long-term dependencies in the data. An LSTM is composed of cells that store information over a long period of time, and can capture and remember patterns that occur in this period. There are 5 key components in each LSTM memory cell: Cell State, Hidden State, Input Gate, Forget Gate, and an Output Gate. The Cell State can be modified, read, and written to, and serves as a long-term memory for the network. The Hidden State is a function of the current network, current cell state, and previous hidden state. This serves as the short-term memory of the network. The Input Gate controls the flow of information in the cell, and decides which input values and previous Hidden State values should be added to the Cell State and updated. The Forget Gate erases any information that is deemed irrelevant. And lastly, the Output Gate is used to determine the next hidden state based on the current input, the updated cell state, and previous hidden state. To help tune this model, we had lowered the number of samples per model from 300 to 5-10. Along with that we had tested different numbers of batch sizes to see which one had performed best.

3.2.3 Bidirectional LSTM

The Bidirectional LSTM is an extension of the LSTM shown previously. The difference is that this model can process input sequences in forward and backwards directions. This allows the model to capture both past and future data. In this model are two LSTM layers for forward and backwards direction. Other features of the model include concatenation and an output layer. The forward LSTM layer processes information from beginning to end, with the backwards LSTM layer processing information in reverse. The concatenation layer combines the information and data from both layers for a single representation for each epoch. Finally, the concatenation is passed through to the output layer where additional processing can occur. This model can detect and recognize patterns that may not be seen by the previous LSTM model because of the forward and backwards processing.

The Bidirectional LSTM had a 91% accuracy with a F1-score of 0.95 normal data, and 0.75 for anomalies. The model is an iteration of the original LSTM that we tested, but we added multiple dropout layers and printed the F1-score for each epoch rather than the accuracy. This was much improved from our previous attempts, as we were now consistently getting over a 0.7 F1-score for anomalies, rather than detecting none to less than 50%. Within the model, we implemented early stopping, a learning rate scheduler, and saved the best model using Tensorboard. We recorded the validation loss for both the train and test data to view where our model was beginning to over fit.

3.3 Implementation

Once our deep learning model identified sequences with artifacts in the data, we then used a signal processing based approach to remove the artifacts. In order to prevent over editing, we only edited the sequences that our model identified as containing anomalies as opposed to adjusting every sequence in a file. We calculated the median and standard deviation of every sequence with anomalies. Within each sequence, we measured each point's distance from the mean, and depending on the point's distance from the median, we either deleted the point or added a certain amount of points to the sequence.

After creating several models to classify what sequences contained anomalies and which did not, we then needed to correct the sequences with anomalies, and put them back into the full IBI data stream. While ideally we would have used a deep learning approach to implement edits, due to time constraints we modified an existing signal processing based approach to identify specific anomaly points and correct the issue.

4 Results

4.1 Detection

This section compares our two detection models with three machine learning models - Random Forest, Support Vector Machine, and Gated Recurrent Unit. The table shows the F1 Score, Anomaly Accuracy, Clean Data Accuracy and the Kappa Score. The F1 Score measures the precision and recall of the model and the Kappa Score measures the inter-rater reliability. For each of these categories, the closer the score is to one the better the model performed.

Model	F1 Score	Anomaly Accuracy	Clean Data Accuracy	Kappa
Random Forest	0.63	0.72	0.89	0.54
SVM	0.42	0.37	0.97	0.39
GRU	0.67	0.71	0.92	0.6
Bidrectional RNN	0.75	0.78	0.94	0.7
LSTM	0.8	0.83	0.9	0.72

Table 2: Results for All Models

Random Forest is an algorithm that uses multiple outputs of decision trees to reach a final result. The algorithm had three different aspects to it which are the node size, number of trees and number of features sampled. Within the model there are decision trees that go through and split the data based on a particular criteria. Table 2 shows the results for the Random Forest. While the model did not perform poorly, we felt as though we were able to get better results

A SVM (Support Vector Machine) is an algorithm that handles linear and nonlinear data within classification problems. The data is classified by finding a hyperplane that maximizes the distance between different points [8]. Our model had used a rbf (radial basis function) kernel to classify our train and test data. When looking at the table, we're able to see that the model performed poorly. This could have been due to having a large data set or due to the difference in features and training.

A GRU (Gated Recurrent Unit) is an algorithm that works best when modeling sequential data. It is composed of 2 gates: an update gate and reset gate, that control the flow of information within the network. We can see that the model performed a bit better than the SVM, but we had other higher performing models.

As discussed earlier in the paper, a bidirectional RNN can process sequences in both forward and backward directions. This is one of the models that had performed the best between the five different ones. We're able to see that this had one of the highest performing F1 scores and anomaly accuracy.

The LSTM was was our best performing model, with the highest anomaly detection performance without sacrificing clean data classification. An LSTM is able to process data from both the past and future and can better retain long term dependencies in the data.

4.2 Performance of Correction Algorithm

In order to measure the success of the editing process, we calculated each raw and edited sequence's RSA value. While the RSA value is not directly a measurement of editing quality, it is a good representation of how constituent the data is throughout. Since edited files have centered data that does not deviate much from the mean, and raw data containing artifacts would be less centered and would result in a higher RSA value, RSA is a good metric to estimate roughly how well an editing algorithm is performing. Additionally, as overedited files would have an even lower RSA value, we also used RSA as an indication of where our algorithm was over editing, implying that it was editing too many points and should be toned down.

5 Discussion

In order to create a baseline for our approach, we initially created a set of traditional Machine Learning and classification algorithms.

Our first algorithm was a Random Forest, which took in each sequence's mean and standard deviation, and would compare each point to those features to classify. This gave us a decent base benchmark to compare to. We also tested other algorithms such as support vector machines (SVMs) as an alternate performance gauge.

Another Deep Learning model we briefly experimented with was the Gated Recurrent Unit (GRU). Since the model performed similarly to the LSTM, we decided to not proceed with it and primarily focus on improving the LSTM's performance.

One issue we ran into was adapting the algorithm to work with our existing data pipeline. As we did not know the original data format the corrections algorithm was designed to handle, we could not adapt our model's output to match the format of the correction algorithm, and instead had to adapt it to work for our specific pipeline. This conversion process initially led to several data processing errors, with very frequent array out of bounds errors and other similar error types. Fortunately, these errors were relatively simple to fix, and involved adding numerous checks to ensure the algorithm was accessing a valid chunk of memory.

Another issue we had more trouble isolating the root cause of was the algorithm not identifying enough points as anomalies. As the correction algorithm consisted of a point check that checked if a point was an anomaly or not, and then ran corrections on respective points, This issue led to our model's detected anomalies not being fixed by the correction algorithm, and to a significant increase in RSA value. We could not determine the root cause of the issue.

Our research has shown that deep learning can be applied to space, and can have relatively good results in the long term to the anomaly detection process. While we feel our model performance could be improved with more complex models, The promising performance we got from even simple models shows that this is one area where DL can be applied to help improve the editing process.

5.1 Future Work

5.1.1 Developing a GUI

While for our purposes a set of scripts worked fine, if anyone wished to actually apply Deep Learning to the space, it would be very beneficial a GUI to make using the models more approachable for those without a coding background. While we tried to document our progress and code as well as we could, it still requires technical know-how to run our current program.

5.1.2 Creating a Deep Learning Network to correct results

While for our purposes a set of scripts worked fine, if anyone wished to actually apply Deep Learning to the space, it would be very beneficial a GUI to make using the models more approachable for those without a coding background. While we tried to document our progress and code as well as we could, it still requires technical know-how to run our current program.

Due to our issues with the RSA data, we did not have time to evaluate a deep learning based approach to correction. Because of how promising our results in the anomaly detection are, we believe that using a similar approach for each sequence detected as containing an anomaly could have good results. Neural networks have been shown to have excellent neural network

6 Conclusion

Throughout the process, we hoped to find whether or not Deep Learning could be applied to the IBI correction process in order to help make the editing process more approachable for beginners, and serve as an additional tool for editors to double check their work and look for any potential issues they may have missed. In addition to anomaly detection, we also seeked to explore automating the anomaly correction process to help convert the labor intensive editing process into one where our application does most of the work, and the editor simply needs to double check the edits done by the model, and fix a few issues that the model may have skipped.

References

[1] Mayo Clinic, "Electrocardiogram (ecg or ekg)." https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983.

[2] E. Burns, "U wave," *Life in the Fast Lane • LITFL*.

[3] F. Shaffer and J. Ginsberg, "An overview of heart rate variability metrics and norms," *Frontiers in Public Health*, vol. 5, p. 258, 2017.

[4] IBM, "What is deep learning?." https://www.ibm.com/topics/deep-learning.

[5] "Event monitor." https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/event-monitor.

[6] S. Porges and R. Bohrer, "Analyses of periodic processes in psychophysiological research," in *Principles of Psychophysiology: Physical, Social, and Inferential Elements*, pp. 708–753, Cambridge University Press, 1990.

[7] S. Porges, "Systems and methods for modulating physiological state," July 7 2020.

[8] IBM, "What is support vector machine?." https://www.ibm.com/topics/support-vector-machine.

[9] IBM, "What are recurrent neural networks?." https://www.ibm.com/topics/recurrent-neural-networks.

[10] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *The European Symposium on Artificial Neural Networks*, 2015.

[11] IBM, "What are convolutional neural networks?." https://www.ibm.com/topics/convolutional-neural-networks.

[12] J. Gamboa, "Deep learning for time-series analysis," *arXiv.org*, 2017.

[13] The MathWorks Inc., "Deep learning for image processing - matlab simulink." https://www.mathworks.com/help/images/deep-learning.html.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436–444, 2015.

[15] N. Rusk, "Deep learning," *Nature Methods*, vol. 13, no. 1, p. 35–35, 2016.

[16] Enterprise Engineering Solutions, "How artificial intelligence is transforming the world?." https://www.eescorporation.com/how-artificial-intelligence-is-transforming-the-world/, Mar. 30 2022.

[17] The MathWorks Inc., "What is deep learning?." https://www.mathworks.com/discovery/deep-learning.html.

[18] B. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *ICLR*, 2020.

[19] Q. e. A. Wen, "Transformers in time series: A survey," *IJCAI*, 2023.

[20] M. R, "Time-series modeling with undecimated fully convolutional neural networks," *stat.ML*, 2015.

[21] chickenbestlover, "RNN-Time-series-Anomaly-Detection." https://github.com/chickenbestlover/RNN-Time-series-Anomaly-Detection.

[22] T. Kaufmann, "ARTiiFACT README." https://github.com/tobias-kaufmann/ARTiiFACT/blob/main/README.md.

[23] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, 2015.

[24] GE Healthcare, "Wearable ecg devices: Considerations for cardiologists." https://www.gehealthcare.com/insights/article/wearable-ecg-devices-considerations-for-cardiologists.