university of groningen / faculty of science and engineering

Annotating ECG signals using echo state networks as a time series classifier

Bachelor's Project Thesis

Lennard de Lange, s3738760, l.k.r.de.lange@student.rug.nl,
Supervisors: Dr F. Cnossen

Abstract: The variability of the interval between two heartbeats, known as the heart rate variability, has been found to be an indicator of many physiological conditions. The methodology of analysis of it consists recording a raw ECG signal and annotating the individual Q-, R-, or S-, peaks within the signal, and then through a discrete Fourier transform on the interbeat interval the power spectrum analysis is executed. The stage of annotating the individual peaks has become partially automated, but still requires manual labour. Threshold based algorithms, such as PreCar , partially automate the annotation of the individual peaks, but such algorithms can be faulty when faced with artifacts in the signal. This paper sets out to assess whether an echo state network is a suitable method to fully automate this process, in comparison to the threshold based algorithm. It was found that the F_1-scores of the echo state network (M = 0.715, SD = 0.054) were significantly lower than the F_1-scores of the threshold based algorithm (M = 0.886, SD = 0.135). Therefore it is concluded that the tested echo state network is not a more accurate tool in automatically annotating ECG signals than PreCar.

1 Introduction

An electrocardiogram (ECG) is a recording of electrical changes in voltage at the skin, due to the contractions in the heart muscles. In this recording it is possible to detect each individual heartbeat, and the timing between two beats, the interbeat interval. Mulder (1992) found that the amount of fluctuations in the interbeat interval, presented as heart rate variability (HRV), is equal to, if not more accurate than, most other physiological indices. Using the HRV, Mulder (1980) found HRV to be strongly correlated to the mental workload, where with increased task load, the heart rate increased, but HRV decreased.

Li et al. (2019) concludes that the most common method of processing HRV is through power spectrum analysis, using fast Fourier transform and autoregressive models, with Sassi et al. (2015) noting more novel methods for HRV analysis. To apply any of these methods, the interbeat interval must be extracted from the ECG data. But as Li et al. (2019) also mentions, this process is thwarted by artifacts, ectopic beats, and arrhythmic events. After correcting for these artifacts, the interbeat interval is extracted by taking the interval between to periodic points in an ECG time series. These periodic points can be either the Q-peak, R-peak, or S-peak. Finding and annotating these points in the ECG time series can be done through multiple ways. One option is through manual labour, this is slow but relatively accurate. A second option is using threshold based software such as PreCar (van Roon & Mulder, 2017), programs like these process the ECG time series by finding where the signal exceeds a set threshold, and automatically annotating them. This is much quicker than manual labour, but also much more prone to mistake due to artifacts in the time series. The software can either miss peaks (false negative), or annotate peaks at points where there is no actual peak (false positive). This results in manual inspection and correction being required, even when using automated software.

This paper sets out to find whether an echo state network (ESN) is a more accurate alternative to threshold systems such as PreCar. An ESN is a specific type of reservoir computing (RC), and encompasses the functionality of a recurrent neural networks (RNN) at a reduced training cost (Lukoševičius, 2012). Connor et al. (1994) found that RNN had advantages over feedfoward neural networks with respect to predicting time series, making them very suitable for matter relating to ECG data. Jaeger (2007) laid out how both a RNN and an ESN are identical in structure (visible in figure 2.1) during the initialisation of the network, but differ when it comes to training the weights between the input layer and hidden layer, the weights between the hidden layers, and the weights between the hidden layer and the output layer. Whereas with a RNN all these weights are adjusted during the training phase, with an ESN only the weights between the hidden layer and output layer are adjusted, and the remaining weights are left at their initialised values. Boedecker et al. (2009) discusses methods of correctly initialising the reservoir as this is imperative to have a functional model without adjusting the weights in the reservoir during the training phase. Tanaka et al. (2019) discuss how RC technology and networks, including ESNs, in part due to their low training cost, are making quick strides in development. This has resulted in exploration of ESNs implemented in other physical systems than traditional computers, which is promising for wearable ECG monitoring devices utilizing ESNs.

RNN are at their core predictive models, and thus using them as a time series classifier (TSC) poses additional challenges. There are mutliple approaches of TSC tasks using an ESN. Lukoševičius and Jaeger (2009) detail a straightforward solution of having real-valued output for each class, creating class probabilities. Ma et al. (2019) adds convolutional and pooling layers after the echo state layer to classify a sequence of points in a time series. Skowronski and Harris (2007) classified time series by having multiple output predictions of each successive point in a time series, and then classifying based on which output had the smallest error to the actual time series. Lukoševičius (2012) covers more ways of classifying a time series, among which is the approach of classifying each point in a time series, aggregating the classification of all the points, and from that reach a final classification of the entire time series.

Sodmann et al. (2018) already found that a convolutional neural network shows significant improvements over threshold based software in annotating ECG data (F_1-score=0.82, 9th overall in the PhysioNet/CinC Challenge 2017), however the used approach required the data to be preprocessed by resampling and signal denoising beforehand.

This paper sets out to answer the question "can an echo state network achieve higher performance rates than a threshold based algorithm on automatically annotating raw ECG signals" and it is hypothesized that the ESN will achieve better performance rates than a threshold based algorithm (i.e. PreCar). The research question will be answered by comparing the F_1-scores of the ESN in correctly predicting the locations of R-peaks in the ECG signal, against the F_1-scores of a threshold based algorithm performing the same task. The employed data pre-processing steps will be limited to normalizing and reshaping the data to make it suitable for the ESN.

2 Methodology

2.1 Dataset

To train and test the ESN, the ECG dataset gathered, annotated, and provided by MD/Phd candidate I.M. Reijmerink will be used. The dataset was collected in 2019 using a Cortrium C3 Holter ECG monitor (sampling rate = 250 Hz), and collected from 28 participants, gender unkown, over the span of two working days. All the participants are surgeons, that during the first day of data collection performed active surgery, and during the second day were present at an outpatient clinic. The data was collected for HRV analysis purposes, and to assess the mental stress and workload of the participants during their surgeon-related duties.

The Cortrium C3 Holter ECG monitors record the ECG signal to BLE files which, by means of the Cortrium Converter Tool, are converted to comma seperated value (.CSV). The .CSV files contain the raw, un-annotated data of three ECG signals, of the same person at the same time. The manual annotation process starts by feeding PreCar the .CSV files, and letting PreCar automatically annotate the desired peaks within the signals, generating event (.EVT) files. After this, the automatic annotation needs a final manual visual inspection and correction of interbeat interval artifacts. The manually corrected data will serve as the ground truth, which will be used for training the ESN and assessing the performance of the ESN and the threshold based algorithm.

Because each ECG signal recording is of varying length, timestamps were noted during the recording of the raw ECG signal, indicating the beginnings and ends of 5 minute blocks. For testing, the dataset exclusively of participant 010 when they were on active operating duty was used, which consists of a 115 five minute blocks. The data of each 5 minute block is then split to segments with a length of 0.12 seconds, containing either a QRS complex, or noise. To train the ESN and assess both the ESN and the threshold based algorithm, the data set was divided in two parts, a training split and testing split. The training split consists of 196959 segments (a randomly sampled selection of 25% of 787837 segments), with 32,6% of the segments containing a QRS complex. The test split consists of 183408 segments, with 27,7% of the segments containing a QRS complex. This means that the training data:test data ratio is roughly 1.07:1.00. These are the final datasets (excluding all data pre-processing required to make the data suitable for the ESN) used for training the ESN, and testing the ESN and PreCar.

2.2 Data pre-processing

To have the ESN be able to process the ECG data, the data must first go through a pre-processing stage. This pre-processing stage consists normalizing the signal in the segments, reshaping the data and the manual annotation into training and test data and targets for the ESN. All the data pre-processing, training and testing the ESN was done within the Python 3.8.10 and require the following packages: EchoTorch 0.2.3, torch 1.10.0, torchvision 0.11.1, numpy 1.21.4, scipy 1.7.3, scikit-learn 1.0.1, sphinx-bootstrap-scm 0.8.0, future 0.18.2, pandas 1.3.5.

The first phase in the pre-processing process is normalizing the signal in the segments. This is done by taking an inverted version of the signal around the x-axis, effectively multiplying all values in a segment by -1. After that, the lowest value, local minima, of the inverted signal is found. Since this is the highest point in the original signal, this is the R-peak in the QRS complex. The signal is then normalized by subtracting the lowest value in the sequence from the entire signal, resulting in the lowest point having a value of 0.

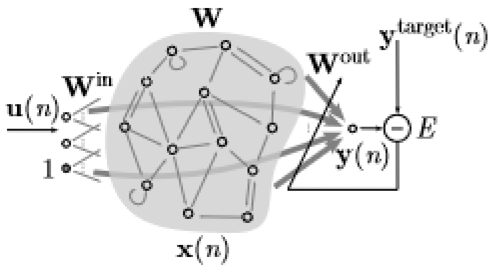

Figure 2.1: The structure of an echo state network (Lukoševičius, 2012)

The second phase in the pre-processing process is to reshape the data segments and the annotation so that they are fit for the ESN. First, the annotation of the peak are turned into two dimensional numpy arrays: [x y] where x is the class probability that the value at that index in the array is noise, i.e. not a R-peak, and y is the class probability that the value at that index in the array is a R-peak. This means that all points have an annotation of [1 0], except for the exact indices where there are R-peaks, these have an annotation of [0 1]. The training/testing data will be reshaped into a one dimensional numpy array with a length of 30, and a corresponding set of target data, which is a two dimensional numpy array containing the class probabilities. Both the data and target arrays are then transferred to torch tensor (multi-dimenstional matrix) which are then passed to the ESN.

2.3 Data exclusion

Earlier research with this dataset has encountered an anomaly with block 6, 7 and 8 from file '5C75A4C8.EVT' during collection of results of the threshold-based algorithm. This prompted a manual inspection of the respective .EVT file, which was found to contain characters that do not appear in any of the other .EVT files, and should not be present in any .EVT file, indicating data corruption. For this reason, block 6, 7, and 8 from file '5C75A4C8.EVT' are excluded from further statistical analysis.

Algorithm 2.1 ESN validation/testing stage

```
Require: $Y_{val}$
Require: $X_{val}$
  for $y_{val} \in Y_{val}$ do
    $X_{pred} \leftarrow$ESN(y$_{val}$)
    for $x_{pred} \in X_{pred}$ do
      if $x_{pred}^{peak} > x_{pred}^{noise}$ and $x_{pred}^{peak} > threshold$ then
        if $x_{val}^{peak} = true$ then
          $case : true\ \ positive$
        else if  then
          $case : false\ \ positive$
        end if
      else if $x_{val}^{peak} = true$ then
        $case : false\ \ negative$
      else if  then
        $case : true\ \ negative$
      end if
    end for
  end for
```

2.4 Model

The model is a leaky-integrated echo state network (ESN), retrieved from the Schaetti (2018) package, with tunable (hyper)parameters. These tunable parameters are: spectral radius, reservoir size, leaky rate, connectivity, and input scaling. The effects and details of these parameters are discussed in section 2.5. An ESN is much like any other neural network, the model is passed on the input data, and the target data. The ESN consists of an input layer, a hidden reservoir, and an output layer. An ESN is a type of recurrent neural network (RNN). The ESN differs from the RNN during the training phase, when all the weights (between the input layer and the reservoir, between hidden layers in the reservoir, and between the reservoir and the output layer) are adjusted. With an ESN only the weights between the reservoir and output layer are adjusted during training (Y(n) in figure 2.1). The remaining weights are left are their initialised values. These initialised values (including the initial value of the weights between the hidden layer and the output layer) are calculated by a normal matrix generator of the Schaetti (2018) package. The input dimension of the ESN is 1 as the input data is a single datapoint, taken from the segment, containing a one dimensional value. The output dimension of the ESN is 2, as the output is a twodimensional array containing the class probabilities of the respective input data point. The class probabilities are then stored in an array, equal in length to the segment from which the input values are taken, and passed on to a comparison algorithm (as visible in algorithm 2.1) to assess whether the segment contains a R-peak, and whether the predictions of individual points are correct with respect to the true labels. This algorithm functions by taking the all the predicted class probabilities (X_{pred}), and then looping through each point (x_{pred}) containing the probability that a point is either a R-peak (x_{pred}^{peak}), or noise (x_{pred}^{noise}). The class probability of a point being a R-peak is compared against the class probability of the point being noise, and against a set threshold. The final value used during testing for the threshold variable is 0.35, and was determined through experimentation. If the class probability of a point being a R-peak passes both comparisons (i.e. x_{pred}^{peak} is larger than x_{pred}^{noise}, and larger than the set threshold), the algorithm compares it against the validation data for that individual point (x_{val}^{peak}).

As mentioned in section 2.1, the model is trained for one epoch on a randomly selected sample of

25% of all data allocated for training, using a least-squares training algorithm. The selection of segments selected for training is determined by generating a pseudo-random distribution, using the built-in-python random module. A desired selection of segments can be obtained by setting a seed before generating the randomly selected sample.

2.5 Hyper-parameter Tuning

The ESN has six parameters, of which are five that can be tuned. The spectral radius, the leaky rate, the reservoir size, the connectivity, and the input scaling. From these five, Lukoševičius (2012) recommends to prioritize the manual tuning of the input scaling, spectral radius, and leaking rate. To optimize the performance of the ESN, these hyper-parameters must be tuned. Hyper-parameters can be tuned based on the nature of the dataset, based on relevant research by others, or through experimentation. These hyper-parameters do not directly dictate the behaviour of the weights, but rather guard the distribution of the weights. This means that the ESN can display unique results with the same hyper-parameters. However, this variation in performance is minimal. The hyper-parameters of the ESN were tuned based on the performance of the ESN while validating on a randomly selected subsample of 25% of the data allocated for training. The ESN is trained on a randomly selected quarter of the training dataset split to further prevent overfitting of the ESN. This is because the training and validation splits are randomized after each epoch of adjusting the hyper-parameters.

The spectral radius hyper-parameter is used in the calculation of the reservoir connection matrix. Lukoševičius (2012) notes the domain of values of the spectral radius hyper-parameter to ensure the echo state property of the ESN, i.e. after a long enough input, the initial state of the reservoir should no longer be recognizable and fully depend on the input series. It is recommended to tune this hyper-parameter based on how long the input should remain in the reservoir. Larger values ensure that inputs are kept in the reservoir for longer, and smaller values result in inputs fading out of the reservoir more quickly.

The leaky rate hyper-parameter is the speed of the reservoir update dynamics discretized in time. This hyper-parameter is advised to be tuned through experimentation, as it is difficult to estimate based on dataset characteristics.

The reservoir size hyper-parameter describes the size of the reservoir. The tuning of this hyper-parameter are task specific, hardware specific, and dataset specific. More complex and challenging tasks require a larger reservoir as that makes it more likely for the reservoir to contain a linear combination of the input signal to approximate the target output. However, too large reservoirs are prone to overfitting, and the computational limits of the hardware (which in turn is task specific again) must be taken into consideration as an upper limit for the reservoir size. A minimum reservoir size can be estimated by taking the number of unique real values of the input that should be stored in the reservoir to approximate the target.

The connectivity hyper-parameter, which is also known as the sparsity of the reservoir, describes how well connected the nodes in the reservoir are to the input layer, to each other, and to the output layer. Although it is advised to keep the connections of the ESN sparse, i.e. a low value for connectivity, this hyper-parameter has a small impact on performance.

The input scaling hyper-parameter determines the factor with which the input matrix scales the input. The input scaling regulates both the amount of nonlinearity represented in the reservoir, and the relative effect the input has on the state of the reservoir. This hyper-parameter can be found by estimating the linearity of the task, but for tasks containing more nonlinear dynamics, experimentation is advised.
The final parameters that were used for the final evaluation of the ESN can be found in table 2.1.

Table 2.1: Used hyper-parameters

spectral radius	0.50
leaky rate	0.90
reservoir size	600
connectivity	0.50
input scaling	0.70

2.6 Evaluating the ESN

The model performance is evaluated by means of calculating the F_1-score (see equation 2.1) for the recognition of R-peaks in the test set. This performance measure will be compared against the F$_1$-score of the threshold-based algorithm used within the PreCar environment.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \quad (2.1)$$

$$precision = \frac{truepositives}{truepositives + falsepositives} \quad (2.2)$$

$$recall = \frac{truepositives}{truepositives + falsenegatives} \quad (2.3)$$

The performance of the ESN starts by pre-processing the testing ECG signal in the same manner the training data is pre-processed (see section 2.2). The final pre-processed segments of ECG data are passed to the ESN for prediction, the predicted annotation of the ESN is then subjected to the comparison algorithm (as visible in algorithm 2.1).

If the model classifies a point in a segment as a R-peak, and the true labels confirms that there is a R-peak at that point in the segment, it is counted as a true positive. If the model classifies a point in a segment as a R-peak, but the true labels does not have a R-peak at that point, it is counted as a false positive. In the case that the model passes through a segment without classifying a single point as a R-peak, but the testing data does indicate that the segment contains a R-peaks, it is counted as a false negative. All other cases result in a true negative, which is the metric with the largest size. The performance of the ESN is assessed by calculating the F$_1$-score per five minute block, over a total of 112 blocks. The total data pipeline used for the ESN and PreCar can be seen in figure 2.2 and 2.3 respectively. These figures also show important differences between the datapipeline used for the ESN and for PreCar. The ESN needs both the training and testing data first to be split up in five minute blocks, which in turn gets split into segments. The segments need to be normalized, before being able to train and test the ESN with the segments. From testing the ESN, a list containing the predicted R-peak locations is then compared against the true labels, from which results are gathered. PreCar takes a single .CSV file containing the raw ECG signal at a time, giving .EVT files containing the predicted R-peak locations. Only then are these .EVT files split up into the same 112 five minute blocks which in turn are compared against the true labels, from which results are gathered.

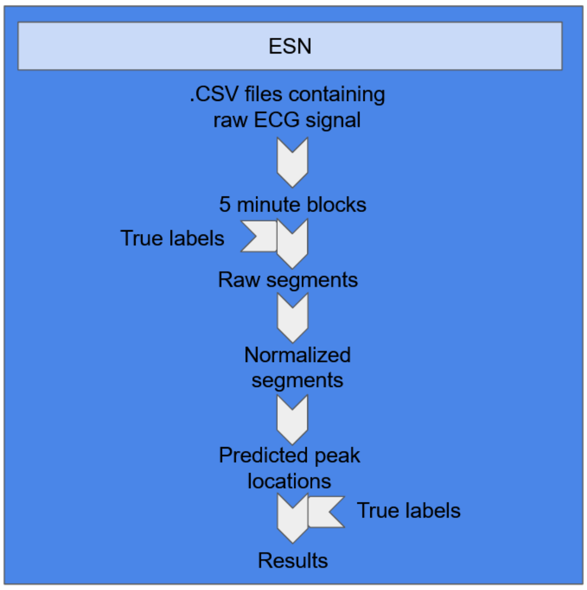

Figure 2.2: The data pipeline used for the ESN

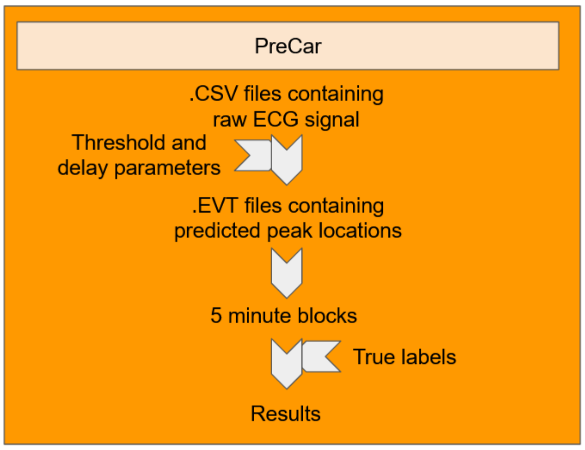
PreCar
.CSV files containing raw ECG signal
Threshold and delay parameters
.EVT files containing predicted peak locations
5 minute blocks
True labels
Results

Figure 2.3: The data pipeline used for PreCar

The performance of the threshold-based algorithm used within the PreCar envorinment is assessed over the same testing data set, distributed over the same five minute blocks and segments. The exact parameters used for threshold-based algorithm are a theshold level of -500 and a delay of 300.

The threshold-based algorithm outputs an .EVT file, containing the timestamps of the suspected R-peaks. These are then split up into the same 112 blocks as for the ESN, and tested against the manually corrected and annotated .EVT file. This testing process follows a method of comparing the timestamps of .EVT file generated by the threshold-based algorithm, and checking whether they coincide with the true labels. These metrics are then used to calculate the F_1-score per five minute block, finally resulting in each five minute block having paired F_1-scores, reflecting the performance of both the ESN and the threshold-based algorithm. To determine whether there is a significant difference between the performance of the ESN and the threshold-based algorithm, a paired t-test is used.

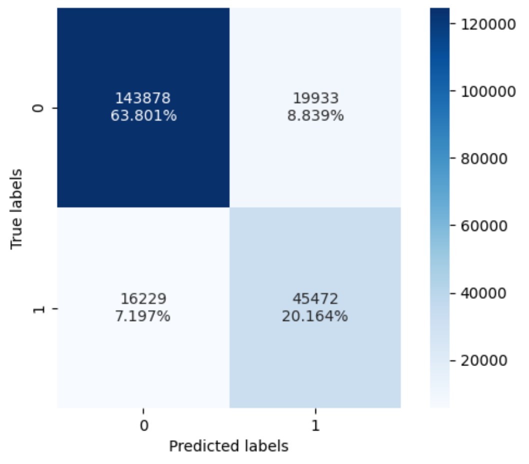

Figure 3.1: Confusion matrix containing the results from the ESN

3 Results

The final results gathered from the ESN from testing a total of 112 five minute intervals of ECG signal. The mean F_1-score of the ESN is 0.652 with a standard deviation of 0.054. The mean F_1-score of the threshold based algorithm is 0.886 with a standard deviation of 0.135. These results can be seen in figure 3.3. All the individual F_1-scores respective to their five minute block, of both the ESN and the threshold based algorithm, can be found in Appendix A, table A.1. Figure 3.1 and figure 3.2 hold the true positive (predicted and validated R-peak), false negative (predicted noise but validated as R-peak), false positive (predicted R-peak but validated as noise), and true negative (predicted and validated noise) numbers of both the ESN and the threshold based algorithm respectively. From these figures, the precision and recall of both the ESN and the threshold based algorithm can be determined. The ESN has a precision of 0.695 and a recall of 0.737 and the threshold based algorithm has a precision of 0.876 and a recall of 0.888. These figures also show that the ESN has higher amounts of both false positives and false negatives than the threshold based algorithm. However, the ESN and the threshold based algortihm are similar in the amount of true positives.

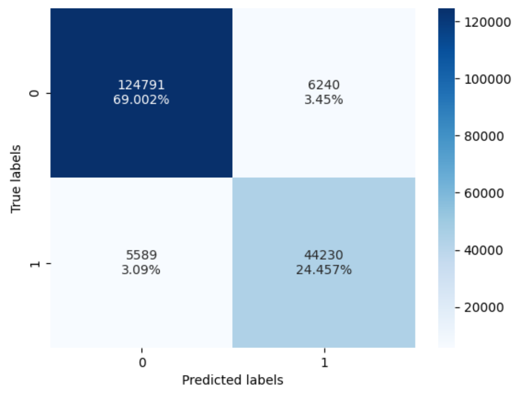

Figure 3.2: Confusion matrix containing the results from the threshold based algorithm

As noted in section 2.3, the mean F_1-scores and standard deviation of both systems have been calculated excluding 6, 7, and 8 from file '5C75A4C8.EVT'. It is found that the F_1-scores of the ESN (M = 0.715, SD = 0.054) are significantly lower than the F_1-scores of the threshold based algorithm (M = 0.886, SD = 0.135), t(112) = -12.388 ($p<.001$).

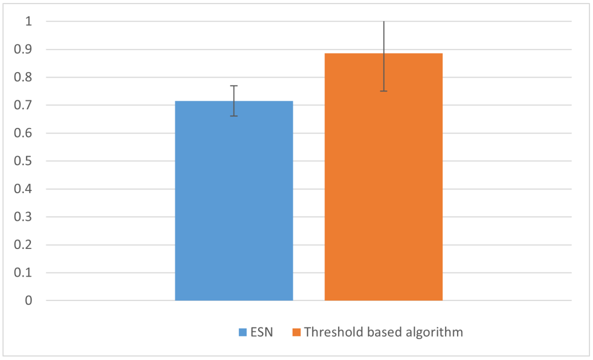

Figure 3.3: F_1-score of both the ESN (blue) and the threshold-based algorithm (orange)

4 Discussion

The developed methodology sets out to employ an ESN to classify R-peaks in a QRS complex from raw ECG signals without needing extensive or manual pre-processing. The ESN and the threshold based algorithm were tested on 112 five minute blocks of ECG signal and achieved a mean F_1-score of 0.715 and 0.886 respectively, which means that the ESN did not yield more accurate results than the threshold based algorithm. From figure 3.1 and 3.2 we can deduce that the ESN and the threshold based algorithm are similar in their true positive rates, but the ESN having higher amounts of both false positives and false negatives than the threshold based algorithm. Since the number of false positive and false negative classifications by the ESN are similar to each other, future improvements on the model must be designed such that the recall of the model improves without having an adverse effect on the precision (e.g. lowering the threshold described in section 2.4 would result in less false negatives but more false positives), and vice versa.

While working out the methodology of this paper, potential issues that could undermine the conclusion were identified. During early exploration of the data, it became apparent that the data quality could be come an limiting factor of the performance of the ESN. For early tuning of the hyper-parameters of the ESN, it is recommended to take a smaller sample of the total training data set to speed up the training process, and be able to adjust the hyper-parameters more quickly. As long as this sample is representative of the entire data set, training on bigger samples should not affect results gravely. However, while initial training started out with just a 1/16 share of the entire data split allocated for training as part of early experimentation, results started to decline harshly after training on more than a 1/4 of the training data split. Since the distribution of the datapoints that were trained on are randomly and evenly distributed across the entire dataset, we can still assume that it is representative (given that the data is heterogenic). However, rare random selections of the training data split would result in very low performance scores. While testing the ESN over multiple random selections of the training data split, it was found that the results discussed in section 3 were not rare or unrepresentative, and thus the designed methodology and the ESN may be valid. An alternative to compensate for the concern of the data quality would be to apply the developed methodology to the standardized MIT-BIH arrhythmia database, to determine whether the low performance can be attributed to the dataset or the model.

A second concern is that the exact shape of the QRS complex is unique on a personal basis, and that the used approach for training and testing possibly undermines the true capability of the methodology. A different training and testing approach could be to train the ESN on a split of a participant's data, and test on the remaining split of the participant's data. Then for the next participant, the ESN would be retrained and re-tested. However, for this approach to be valid, it would require enough data of each participant to adequately train and test the ESN each time, which could result in unrepresentative results in case the participants' do not have enough data on an individual basis. For that reason, this methodology opted for classifying the data of one participant with a model trained on the data of multiple other participants. Future endeavours that employ the methodology laid out in this paper but focus on a personalized dataset, could determine whether unique personalized features in a QRS complex has an effect on the performance of the model.

A third point of concern is the methodology used for training and validating the ESN. As noted in section 2.1, the model is first trained on a random selection of 25% of all the data allocated for training, and then validated of another random selection of 25% of all the data allocated for training. It is unavoidable that these two unique selections contain the same datapoints, which might cause overfitting. A possible approach to counteract this could be to validate the ESN on the data of a participant on which it will never train. However, this in turn might cause overfitting to that dataset (and to an extent, the QRS complex of that participant), thus shifting the problem instead of solving it. This could be solved by validating on fewer samples of the participant, and randomizing the selection of those samples per epoch.

5 Conclusions

This paper set out to assess whether an ESN is a suitable tool to automatically annotate ECG signal without the need for extensive or manual pre-processing of the signal. The performance of the ESN, measured in an average F_1-score, was compared to that of threshold based algorithm found within PreCar. It was hypothesized that the ESN would outperform the threshold based algorithm. From the results it was found that the F_1-scores of the ESN were significantly lower than the F_1-scores of the threshold based algorithm. This means that we have to reject our hypothesis, and thus conclude that this implementation of an ESN does not outperform the threshold based algorithm. However, even though the ESN did not outperform the threshold based algorithm, it still showed promising results. Further fine tuning of the hyperparameters, or utilising a more powerful low level architecture could lead to an automated ESN-based ECG annotation tool.

References

Boedecker, J., Obst, O., Mayer, N. M., & Asada, M. Studies on reservoir initialization and dynamics shaping in echo state networks. In: *Esann*. Citeseer. 2009.

Connor, J., Martin, R., & Atlas, L. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, *5*(2), 240–254. https://doi.org/10.1109/72.279188

Jaeger, H. (2007). Echo state network. *scholarpedia*, *2*(9), 2330.

Li, K., Rüdiger, H., & Ziemssen, T. (2019). Spectral analysis of heart rate variability: Time window matters. *Frontiers in neurology*, *10*, 545.

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (pp. 659–686). Springer Berlin Heidelberg.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Ma, Q., Zhuang, W., Shen, L., & Cottrell, G. W. (2019). Time series classification with echo memory networks. *Neural Networks*, *117*, 225–239.

Mulder, G. (1980). The heart of mental effort. *Studies in the cardiovascular psychophysiology of mental work. Doctoral dissertation. University of Groningen, Groningen, The Netherlands.*

Mulder, L. (1992). Measurement and analysis methods of heart rate and respiration for use in applied environments [Special Issue Cardiorespiratory Measures and thier Role in Studies of Performance]. *Biological Psychology*, *34*(2), 205–236. https://doi.org/https://doi.org/10.1016/0301-0511(92)90016-N

Sassi, R., Cerutti, S., Lombardi, F., Malik, M., Huikuri, H. V., Peng, C.-K., Schmidt, G., Yamamoto, Y., Reviewers: D., Gorenek, B., Lip, G. Y., Grassi, G., Kudaiberdieva, G., Fisher, J. P., Zabel, M., & Macfadyen, R. (2015). Advances in heart rate variability signal analysis: joint position statement by the e-Cardiology ESC Working Group and the European Heart Rhythm Association co-endorsed by the Asia Pacific Heart Rhythm Society. *EP Europace*, *17*(9), 1341–1353. https://doi.org/10.1093/europace/euv015

Schaetti, N. (2018). Echotorch: Reservoir computing with pytorch.

Skowronski, M., & Harris, J. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, *20*(3), 414–423.

Sodmann, P., Vollmer, M., Nath, N., & Kaderali, L. (2018). A convolutional neural network for ecg annotation as the basis for classification of cardiac rhythms. *Physiological measurement*, *39*(10), 104005.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., & Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, *115*, 100–123.

van Roon, A., & Mulder, L. (2017). Precar version 3.7 user's manual version 1.9.

A Appendix

Table A.1: The F-scores of the ESN and the threshold based algorithm

filename	block number	F-score ESN	F-score threshold based algorithm
5C74ECD8.CSV	1	0.77122	0.99031
5C74ECD8.CSV	2	0.73019	0.99708
5C74ECD8.CSV	3	0.68768	0.99371
5C74ECD8.CSV	4	0.77321	0.99842
5C74ECD8.CSV	5	0.75000	0.99000
5C74ECD8.CSV	6	0.77054	0.99197
5C74ECD8.CSV	7	0.76923	1.00000
5C74ECD8.CSV	8	0.56386	0.92623
5C74ECD8.CSV	9	0.64320	0.99404
5C74ECD8.CSV	10	0.74458	0.95466
5C75A4C8.CSV	1	0.74405	0.98666
5C75A4C8.CSV	2	0.72624	0.87292
5C75A4C8.CSV	3	0.72298	0.93591
5C75A4C8.CSV	4	0.71638	0.75514
5C75A4C8.CSV	5	0.72129	0.74051
5C75A4C8.CSV	6	0.69468	0.01855
5C75A4C8.CSV	7	0.70485	0.00705
5C75A4C8.CSV	8	0.72530	0.04192
5CC93EE3.CSV	1	0.77098	0.99750
5CC93EE3.CSV	2	0.71949	0.85319
5CC93EE3.CSV	3	0.71317	0.98882
5CC93EE3.CSV	4	0.67526	0.93224
5CC93EE3.CSV	5	0.73639	0.94031
5CC93EE3.CSV	6	0.73940	0.99155
5CC93EE3.CSV	7	0.65233	0.95471
5CC93EE3.CSV	8	0.64495	0.91236
5CC93EE3.CSV	9	0.74175	0.97058
5CC93EE3.CSV	10	0.71602	0.94689
5CC93EE3.CSV	11	0.68341	0.96936
5CC93EE3.CSV	12	0.72421	0.97371
5CC93EE3.CSV	13	0.75632	0.99303
5CC93EE3.CSV	14	0.74175	0.99412
5CC93EE3.CSV	15	0.70909	0.97549
5CC93EE3.CSV	16	0.71582	0.92325
5CC93EE3.CSV	17	0.69976	0.95945
5CC93EE3.CSV	18	0.71848	0.97111
5CC93EE3.CSV	19	0.74383	0.98461
5CC93EE3.CSV	20	0.78800	0.99059
5CC93EE3.CSV	21	0.73567	0.98071
5CC93EE3.CSV	22	0.75303	0.97400
5CC93EE3.CSV	23	0.74975	0.95045

Table A.1: continued from previous page

filename	block number	F-score ESN	F-score threshold based algorithm
5CC93EE3.CSV	24	0.70478	0.97520
5CC93EE3.CSV	25	0.74166	0.97520
5CC93EE3.CSV	26	0.74894	0.97963
5CC93EE3.CSV	27	0.72553	0.87605
5CC93EE3.CSV	28	0.76202	0.95015
5CC93EE3.CSV	29	0.76701	0.91772
5CC93EE3.CSV	30	0.71285	0.86855
5CC93EE3.CSV	31	0.75383	0.93248
5CC93EE3.CSV	32	0.70669	0.85482
5CC93EE3.CSV	33	0.68915	0.88056
5CC93EE3.CSV	34	0.73642	0.86591
5CC93EE3.CSV	35	0.71281	0.83054
5CC93EE3.CSV	36	0.71883	0.88959
5CC93EE3.CSV	37	0.72222	0.81673
5CC93EE3.CSV	38	0.69216	0.86424
5CC93EE3.CSV	39	0.69614	0.88659
5CC93EE3.CSV	40	0.74147	0.83603
5CC93EE3.CSV	41	0.73580	0.96963
5CC93EE3.CSV	42	0.77788	0.94545
5CC93EE3.CSV	43	0.76600	0.96982
5CC93EE3.CSV	44	0.71204	0.91173
5CC93EE3.CSV	45	0.73452	0.96436
5CC93EE3.CSV	46	0.68537	0.99475
5CC93EE3.CSV	47	0.68186	0.94079
5CC93EE3.CSV	48	0.75665	0.97007
5CC93EE3.CSV	49	0.73664	0.98996
5CC93EE3.CSV	50	0.77454	0.91053
5CC93EE3.CSV	51	0.78389	0.82109
5CC93EE3.CSV	52	0.68500	0.79731
5CC93EE3.CSV	53	0.74297	0.85018
5CC93EE3.CSV	54	0.69094	0.96296
5CC93EE3.CSV	55	0.74745	0.97320
5CC93EE3.CSV	56	0.73526	0.96907
5CC93EE3.CSV	57	0.71901	0.94478
5CC93EE3.CSV	58	0.70188	0.89252
5CC93EE3.CSV	59	0.72233	0.95351
5CC93EE3.CSV	60	0.72294	0.92864
5CC93EE3.CSV	61	0.77272	0.97689
5CC93EE3.CSV	62	0.74928	0.97492
5CC93EE3.CSV	63	0.75098	0.91609
5CC93EE3.CSV	64	0.68686	0.87649
5CC93EE3.CSV	65	0.74799	0.91050
5CC93EE3.CSV	66	0.74751	0.86586
5CC93EE3.CSV	67	0.59238	0.78178

Table A.1: continued from previous page

filename	block number	F-score ESN	F-score threshold based algorithm
5CC93EE3.CSV	68	0.62379	0.75939
5CC93EE3.CSV	69	0.71209	0.75963
5CC93EE3.CSV	70	0.74657	0.76294
5CC93EE3.CSV	71	0.77256	0.77389
5CC93EE3.CSV	72	0.65139	0.77959
5CC93EE3.CSV	73	0.68405	0.87194
5CC93EE3.CSV	74	0.70329	0.93251
5CC93EE3.CSV	75	0.62882	0.87984
5CC93EE3.CSV	76	0.66171	0.59725
5CC93EE3.CSV	77	0.66666	0.33237
5CC93EE3.CSV	78	0.68487	0.61325
5CC93EE3.CSV	79	0.67329	0.92579
5CC93EE3.CSV	80	0.78250	0.70985
5CC93EE3.CSV	81	0.73985	0.86385
5CC93EE3.CSV	82	0.69071	0.69152
5CC93EE3.CSV	83	0.72268	0.87527
5CC93EE3.CSV	84	0.73800	0.93501
5CC93EE3.CSV	85	0.78396	0.87148
5CC93EE3.CSV	86	0.76527	0.74027
5CC93EE3.CSV	87	0.73990	0.64
5CC93EE3.CSV	88	0.71229	0.63551
5CC93EE3.CSV	89	0.69988	0.62803
5CC93EE3.CSV	90	0.72572	0.66737
5CC93EE3.CSV	91	0.70071	0.66807
5CC93EE3.CSV	92	0.66062	0.96875
5CC93EE3.CSV	93	0.69747	0.88396
5CC93EE3.CSV	94	0.69836	0.98748
5CC93EE3.CSV	95	0.67452	0.94868
5CC93EE3.CSV	96	0.51418	0.98519
5CC93EE3.CSV	97	0.39613	0.98176