Detecting arrythmia using ECG RR intervals and Deep learning method

Abstract

Atrial fibrillation, either permanent or intermittent increases the risk of cardioembolic stroke. In order to prevent and treat effectively, one must provide an accurate diagnosis. Long term ECG monitoring improves the likelihood of diagnosing AF. In this project, we implemented a Bidirectional Recurrent Neural Network using LSTM layers to detect AF in long ECG recordings. After comparing results with the research article from Faust et al. [1], we experiment different scenarios to ensure a balance and clean data as well as tested different hyperparameters and a replacement for LSTM cells to improve their results. We achieved an accuracy of 85% in the test set when reproducing the exact same condition as in the article. Although we couldn't improve their accuracy result, we explored various directions and observed the importance random samples when using overlapping sequences with LSTM in order to ovoid overfitting.

Introduction:

Heart arrhythmias or heart rhythms problems occurs when the electrical impulses that coordinate your heartbeats do not work properly, causing your heart to beat too fast, too slow or irregularly [1] as we can see in Figure 2 where the electrocardiogram (ECG) shows an irregular heart beat rhythm. An ECG is a graph of voltage versus time of the electrical activity of the heart using electrodes placed on the skin. These electrodes detect the small electrical changes that are a consequence of cardiac muscle depolarization followed by repolarization during each cardiac cycle (heartbeat). Changes in the normal ECG pattern occur in numerous cardiac abnormalities, including cardiac rhythm disturbances (such as atrial fibrillation and ventricular tachycardia), inadequate coronary artery blood flow (such as myocardial ischemia and myocardial infarction), and electrolyte disturbances (such as hypokalemia and hyperkalemia) [7]. There are three main components to an ECG: the P wave, which represents the depolarization of the atria; the QRS complex (as seen in Figure 1), which represents the depolarization of the ventricles; and the T wave, which represents the repolarization of the ventricles.

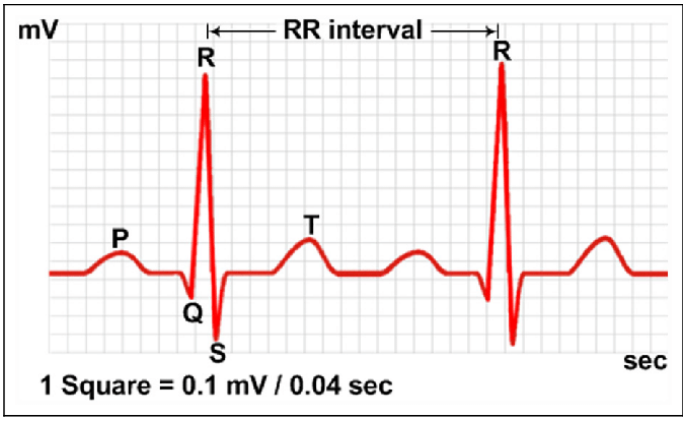


Figure 1 ECG waveform and RR interval

Atrial Fibrillation (AF), the most common arrhythmia, arises from irregular atrial contraction. This disease affects around 0.4% of the adult population and it gets more prevalent with age. Some patients with AF are

asymptomatic, but others have symptoms like fainting, palpitations, chest pains and heart failure which diminish substantially patients' quality of life. AF is associated with inefficiencies in blood flow dynamics which increase the risk of stroke and systemic thromboembolism, resulting in high mortality and morbidity. There is thus clinical value in developing a method to automatically detecting AF.

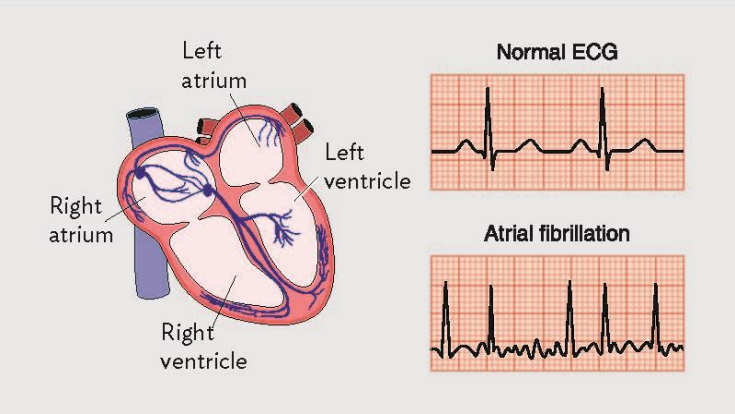


Figure 2 Heart schema and comparison between normal ECG and Atrial Fibrillation

Over the past decades, research has shown that neural networks and deep learning algorithms can be used to detect AF from continuous electrocardiogram recordings. However, automatic arrhythmia classification faces several important challenges [3]. For example, arrythmia symptoms might not be seen during the ECG signal capturing period, ECG signals can very from person to person and depends on factors such as age, gender physical conditions and lifestyle which complicates the generalization of the model. Also, the volume of data to be considered for ECG signal analysis is large which increase the possibility of false diagnosis. Finally, noise artifacts and interference can result in morphological variations in the ECG signal. In the last few years, various different deep learning methods have been used on ECG signals for the classification purpose and to overcome those challenges. For example, one research used a Deep Belief Network to detect and differentiate poor from high quality ECG signals in order to improve afterwards the results in arrhythmia detection [4].

Recurrent neural networks have gained a lot of popularity in this field because they allow the network to retain and use state informations (informations about what has happened in previous time steps). In 2017, Zhang et al. used an RNN to learn time correlation of ECG signal points. Morphology information of the ECG signal including the T wave of former and present beat were given as input into the RNN. They obtained in the detection of Ventricular and Supraventricular ectopic beat an accuracy of 99.4% and 98.7% respectively. In 2018, Yildirim proposed a new method using LSTM and a new wavelet-based layer that decomposed the ECG signal into frequency sub bands at different scales. This wavelet layer was used to generate the ECG signal sequences. Their model succeeded in the classification of ECG signals into five different heartbeat types and their results showed high recognition performance: 99.39%. Recurrent neural networks (RNN, LSTM and GRU) methods showed high accuracy as well in atrial fibrillation (AF) classification [3]. AF is associated with an ECG that has irregular R-R intervals (time between two R peaks) [4]. Hence, a majority of AF detection algorithms tries to detect such irregularities in those intervals to predict AF. This is the case with [1], an article from Faust et al. that uses as inputs 100 beats of RR intervals and fed them in a Bidirectional Recurrent Neural Network with LSTM layers to classify each beat sequency as AF or non-AF.

In this project, we aim to develop a model that can be used to detect AF from long ECG recordings and thus provide a robust diagnosis support system for AF. We will use the MIT-BIH Atrial Fibrillation dataset [2]. It contains 25 long term ECG recordings of human subjects with AF. The problem will be tackled in a similar manner as [1]: a classification Bidirectional Recurrent Neural Network (BRNN) that will get RR intervals as input and will predict if an atrial fibrillation is present or not. The data was preprocessed and split into overlapping 100 beats of RR intervals.

A BRNN model was implemented and trained as a classification problem. One issue we faced with this article is that when using non continuous data as they did, we get a an 85% accuracy but we have no use for LSTM layers since no sequence and no memory are required but when we do use continuous data, we face overfitting and thus low accuracy in the validation set. In order to overcome this issue, we experiments various scenarios playing with hyper-parameters, adding and removing layers, adding dropouts, removing LSTM layers to compare it s usefulness and switching it with GRU layers. Another issue in [1] is that the data is unbalanced with mostly beat sequence without AF. To solve this issue, we tried oversampling methods to increase the number of AF sequence and balance the data.

Methods:

Database:

The database used is MIT BIH Atrial fibrillation from Physionet. [5] It includes 25 long term ECG recordings of human subjects with atrial fibrillation (mostly paroxysmal). Of these 25 recordings, two signals: 03665 and 00735 are represented only by the rhythm and don t contain the two ECG signals. All 23 other recordings are each 10 hours in duration, contains two ECG signals each sampled at 250 samples per second with 12-bit resolution over a range of +- 10mV. Those recordings were made at Boston's Beth Israel Hospital using ambulatory ECG recorders with a bandwidth of approximately 0.1Hz to 40Hz.

This database contains rhythm annotations files (with suffix .atr files) prepared manually of types AFIB for atrial fibrillation, AFL for atrial flutter, J for AV junctional rhythm and N for all other rhythms. It also contains beat Annotations files with suffix .qrs that were calculated using an automated detector. However some patients have manually corrected beat annotations like the patients 05091 and 07859 with suffix .qrsc.

For the patients 04043, 08434 and 08405, some blocks of the ECG signal was unreadable so the missing data blocks of duration 10.24s were replaced with a flat segment of samples of amplitude zero (Figure 3).

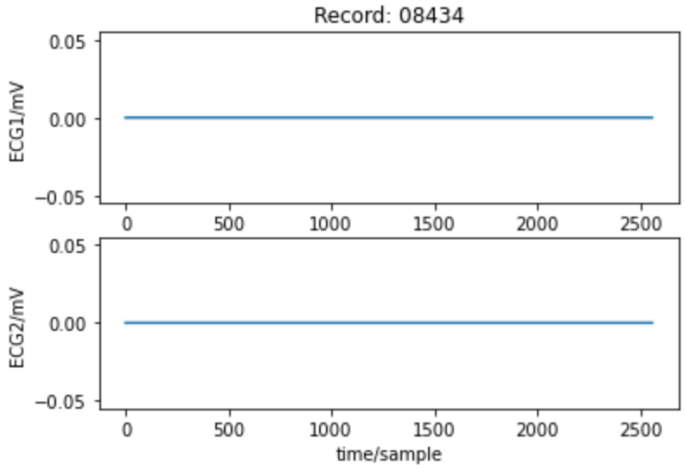


Figure 3 Flat segment replacing missing data in patient 0843

Signal Processing:

- Data Exploration

  The ECG of each patient was first analyzed manually in order to understand better the data, how noisy it was, where was the missing data. To read and analyze the ECG recordings, the wfdb python library was used. We can see in Figure 4 that the R peaks are labelled (red dots)

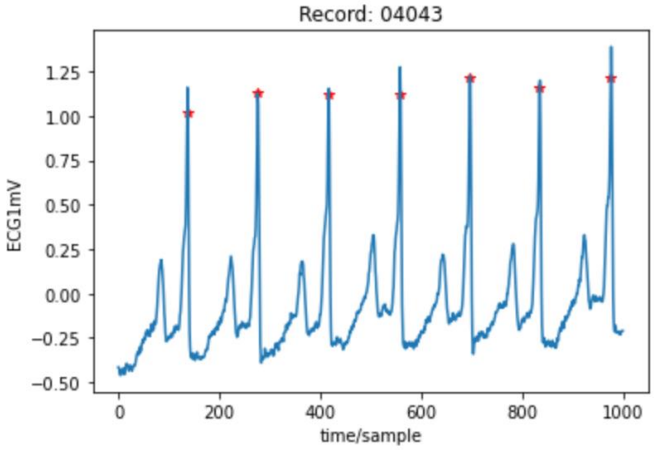


Figure 4 record sample of a patient from MIT BIH Atrial Fibrillation database

- Data building and cleaning

  We have decided to limit to signal processing to its minimal in order to provide a robust model capable of giving accurate results using mostly raw data. Furthermore, the decision-making system gets all the information with the data blocks and we did not extract any extra features. Based on the R peaks labels, the RR intervals were extracted and split into overlapping sequences of 100 beats for each HR trace as done in [1]. We labelled each beat sequence as AF (atrial fibrillation) if one or more RR interval were classified as AF in the database annotations.
  At this point, with raw data, the number of AF labels is 438022 and the number of non-AF labels is 530281. Afterwards, the data was cleaned, and outliers were extracted in the following method: The interval length was converted into heart rate and every heart rate bigger than 200 and smaller than 50 were removed. At each step we calculated the ratio between AF labels and non-AF labels in order to prevent unbalancing the data. At the end of data cleaning, 929281 were non-AF labels and 26680 were AF labels. Consequently by removing outliers and preprocessing the data we increased significantly the ratio between AF and non-AF labels.
  Various methods to balance the data at the preprocessing step were used like Smote [5] and weren't efficient since there is a link between every beat sequence of a patient due to the overlapping.

Model from [1]:

In the article that was implemented [1], they proposed a Bidirectional Recurrent Neural Network (BRNN) using LSTM blocks to classify RR intervals into 2 categories: AF or no-AF. The advantage of a BRNN is that it can simultaneously get information from past and future states of a sequence by connecting two hidden layers of opposite directions to the same output. In their architecture (Figure 4), the number of LSTM cells in each of the forward and backward layers was twice the input sequence length with two fully connected layers at the top of the model to produce the final classification results. They also used a global max pooling between the LSTM layers. This bidirectional LSTM layers were used to extract features from the Heart Rate data sequence before passing them to the fully connected layers.

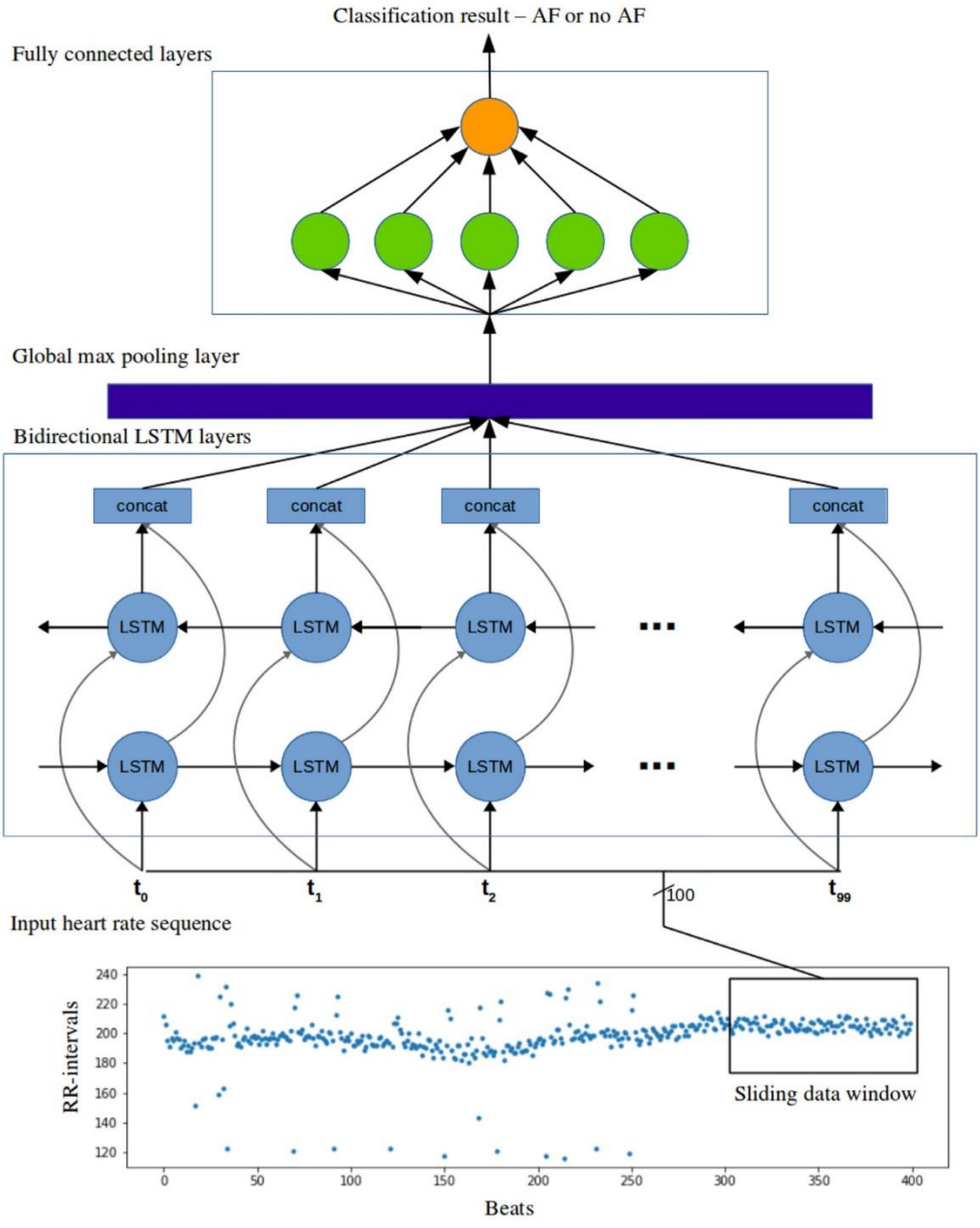


Figure 5 The bidirectional LSTM architecture used for AF classification

The weights were initialized using Xavier initialization and Adam optimizer was used for the gradient descent backpropagation to update the weights. The initial learning rate was set to 1e-3 and a binary cross-entropy function was used to evaluate the loss of the network. Batch size was set to 1024 in order to provide a good trade off between available GPU memory and speed of training. Furthermore, they applied a Recurrent dropout during the training of inputs and hidden states of LSTM cells and standard dropout was applied between fully connected layers to reduce the overfitting of the model and improve generalization [1]. Finally, they performed a cross validation to evaluate their model performance and tune the model and hyperparameters.

Table 1
Bidirectional Long Short-Term Memory (LSTM) architecture.

Layer	Type	Output shape	Number of parameters
1	Input	100, 1	0
2a	LSTM (forward)	100, 400	161600
2b	LSTM (backward)	100, 400	161600
3	Global 1D max pooling	400	0
4	Fully connected Rectified Linear Unit (ReLU)	50	20050
5	Dropout	50	0
6	Fully connected (Sigmoid)	1	51

The article provided a repository as well with their data and the code used. We first reproduced the same model using Pytorch with the same parameters and compared results (see section Experiments and Results).

When looking at their code and their data we noticed that they were using non continuous data. In other terms, when loading their data during the training, they are taking samples randomly and thus there is no connections between consecutive samples even though they took overlapping beats sequence in order to create a link between sequences. In order to validate our intuition, we did two experiments: one where we remove the random samples and use continuous data and another one where we removed the LSTM cells and use non continuous data. When using continuous data, we faced important overfitting and tried to reduce it by adding some layers, adding dropout and also by replacing LSTM cells with GRU. When removing LSTM cells, we obtained high accuracy and low loss (see section Experiments and Results).

Implementations and Experiments:

We implemented a BRNN with the exact layers and parameters as described in [1]. To do so, we created a Bidirectional class inherited from nn.Module that initialized and set hidden dimensions to 200 and set the LSTM layers with input of a 100 (100 beats for each sequence), hidden size as 200, dropout configured as hyperparameter set as 0, bias set as True, bidirectional parameter set as True and num layers configured as hyperparameter set as 1.
We then initialized a MaxPooling1d layer with hyperparameter 1, a fully connected layer with input 400 and output 50, a ReLU activation function, a dropout layer with hyperparameter 0.1, another fully connected layer with input 50, output 1 and finally a sigmoid activation function. The loss used was BinaryCrossEntropy loss and the optimizer chosen was Adam.
As data, we used the preprocessed data that went through all the steps explained earlier in the section Methods. The data was first split into train and test set. Twenty patients for the training set and 3 patients for the test set. Afterwards we split the training set into training and validation set to evaluate the performance. Batch size was set to 1024 as advised in [1] and epoch set to 80.
We implemented our own sampler inherited from sampler class and define a dataloader that would iterate through our samples. The samples would have 100 beats + 1 label at the end that would have value 0 for no-AF and 1 for AF.

Experiment 1:

This first experiment was to compare our model with the model implemented in [1] and to evaluate differences due to the signal processing steps. After training our model, we tested it and evaluated the loss and the accuracy (see in section Results)

Experiment 2:

In the Second experiment, we used raw data, removed all the preprocessing steps and just fed the model with sequences of 100 beats of raw data. We kept the same hyperparameters, batch size, samples (see in section Results). This experiment was done in order to check the impact of signal processing in the model training. Also, it is a way to improve our result from experiment 1 where although the accuracy was elevated, the model only predicted 0 and never predicted 1.

Experiment 3:

The same configuration as experiment 2 was set up. However, we changed some hyperparameters to improve the results. We add dropout in the LSTM cells with hyperparameter 0.2 in order to reduce overfitting and we also reduced the learning rate to 0.0001 for the same reason. Batch sized stayed set at 1024.

Experiment 4:

For the fourth experiment, we removed the LSTM cells but kept every other layer described in Experiment 1 and used raw data as input. The point of this experiment was to compare with the results of experiment 2 and check if the LSTM cells was indeed necessary if the data used is non continuous.

Experiment 5:

For the fifth experiment we removed the sampler so that the Dataloader would take the sequences one after another and thus sequences would be linked because of the overlapped input. We took the same configuration as experiment 2 with raw data to compare result between the two experiments: one with non-continuous data and one with continuous data.

Experiment 6:

Same as experiment 5 but this time with our cleaned data to check if with continuous data the model would learn better how to predict AF even though the data is unbalanced.

Experiment 7:

In order to reduce the overfitting observed during experiment 5, we took the same configuration as experiment 5 but tried an oversampling method using SMOTE from Sklearn library. Using our cleaned data, we obtained with oversampling: 85420 AF and 941137 for non AF. We applied it on the training set and thus tried to have an impact on the unbalanced data which explains the overfitting.

Experiment 8:

In this experiment we returned to non continuous data which gave the best results but tried to improve it by replacing LSTM cells with GRU. The GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU's got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate. We used the exact same parameters as experiment 1 or 2 with raw data.

Results and discussion:

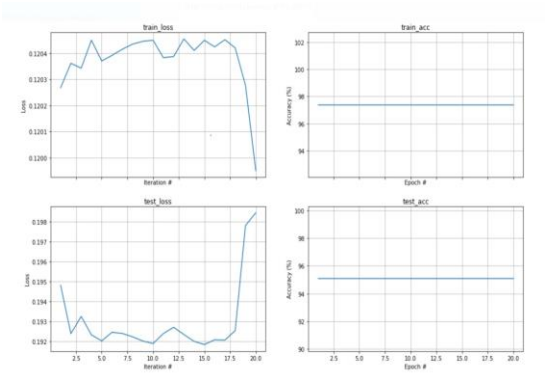
Experiment	Loss in test set	Accuracy in test set	Explanation
1	0.129	97%	We can see in this experiment overfitting so the loss and accuracy in the test set seems good but when we look at the confusion matrix we can see that the model is only predicting label 0 because of the unbalanced data and have good accuracy because most of the data is indeed with label 0.
2	0.701	85%	Those result are coherent with the data. Indeed, we used the raw data that was more balanced, thus during the training the model was better trained to recognize AF and thus did predict right during the test and wasn't always predicting 0 as in exp 1
3	3.13	60%	We tried some different hyperparameters on the raw data to improve our results from experiment 2 adding dropout and layers but it didn't improve the accuracy. Here is an example of results we obtained.

4	5.38	70%	This experiment removed the LSTM cells with non continuous raw data. Looking at the confusion table we observed that the model predict a high rate of false positive which means than for a high number of samples, it predict 0 instead of 1 and thus predict non AF instead of AF. Comparing with experiment 2, we understand that although the data is not continuous, Bidirectional LSTM is still necessary to obtained acceptable results
5	0.7	54%	When removing the sampler and thus using continuous data even with raw data we observed significant overfitting The overfitting is caused by the data that is overlapped. Indeed, every beat sequence of a patient has 99 beats common with the precedent sequence and one beat changing. Consequently, BRNN having access to past and future output could overfit while training.
6	0.4	91%	Here we used the oversampling with SMOTE to add some AF samples in our cleaned data. In the results we can see a high accuracy but the oversampling was not efficient and the model only predicts non-AF
7	2.43	55%	The results here may seem worse than for experiment 1 where we also used our cleaned data but we can see this time that the model is predicting AF even though

			the accuracy is low. The problem with using continuous data is overfitting as we can see in the plots of exp6 where the training accuracy is high on the contrary of the test accuracy which is constant. There is the same problem as experiment 4
8	1.88	57%	In this experiment, we replaced LSTM with GRU another long-term memory neural network. We observed a lowest accuracy than with LSTM cells.

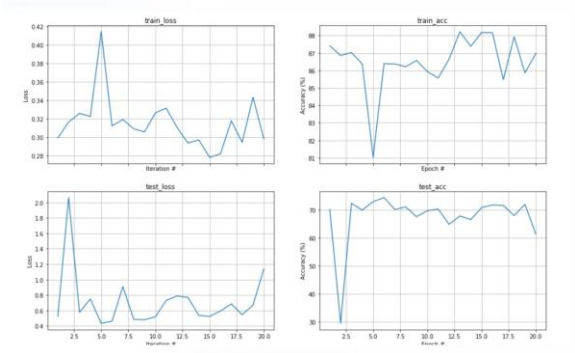
- *Plots and confusion matrix for experiments:*

1. *Exp 1:*



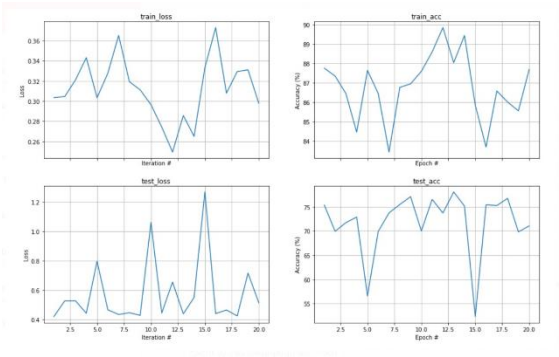
Confusion table	*True*	*False*
Positive – non-AF	*929281*	*26680*
Negative – AF	*0*	*0*

2. *Exp 2:*



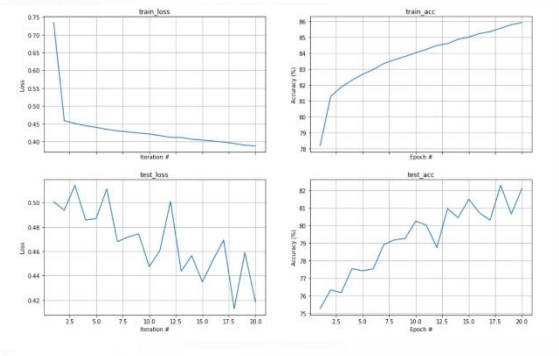
Confusion table	*True*	*False*
Positive – non-AF	*441548*	*48429*
Negative – AF	*389593*	*88733*

3. *Exp 3:*



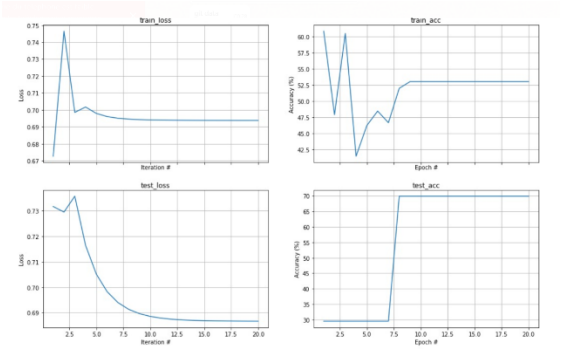
Confusion table	*True*	*False*
Positive – non-AF	*347168*	*90854*
Negative – AF	*239052*	*291229*

4. *Exp 4:*



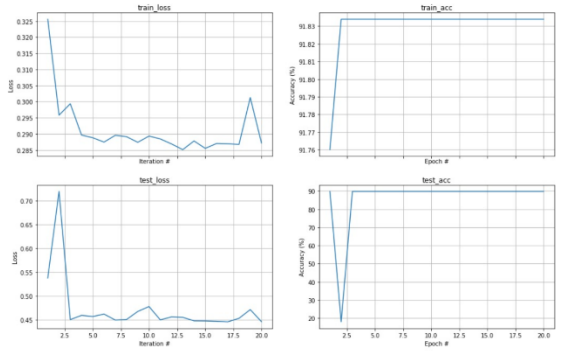
Confusion table	*True*	*False*
Positive – non-AF	*511178*	*272923*
Negative – AF	*165099*	*19103*

5. *Exp 5:*



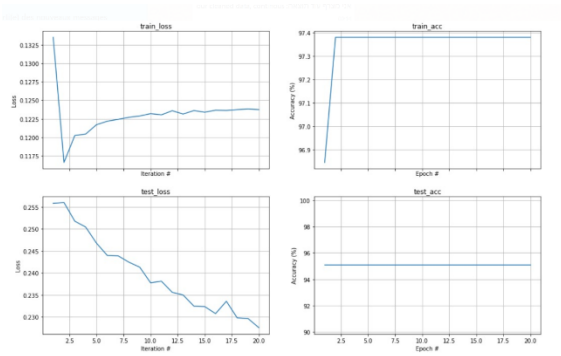
Confusion table	*True*	*False*
Positive – non-AF	*530281*	*438022*
Negative – AF	*0*	*0*

6. *Exp 6:*



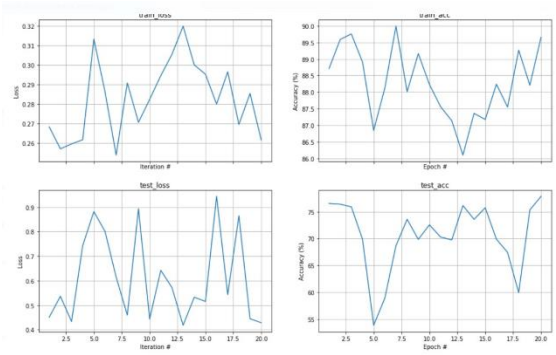
Confusion table	*True*	*False*
Positive – non-AF	*941137*	*85420*
Negative – AF	*0*	*0*

7. *Exp 7:*



Confusion table	*True*	*False*
Positive – non-AF	*506670*	*5294*
Negative – AF	*21386*	*422611*

8. *Exp 8:*



Confusion table	*True*	*False*
Positive – non-AF	*506251*	*397573*
Negative – AF	*50449*	*24030*

Conclusion:

To conclude, we aim in this project to reproduce the experiment [1] with a BRNN model in order to classify ECG sequences into AF or non-AF. We implemented their model respecting their number of layers, hyperparameters, batches size and epoch and obtained an accuracy of 85% on the test set. However, while cleaning and removing some outliers during the preprocessing step, we realized that removing outliers unbalanced the data and that even with oversampling methods like Smote, we couldn't overcome the unbalance data and thus train the model well enough for it to predict accurately AF. We then used only raw data which was more balanced to overcome this issue. We also made some experiments to understand the importance of using non continuous data instead training the samples one after another and thus have during training at each sample only one different beat from the previous one. We observed that using the data continuously provoked a significant overfitting during training and consequently a low accuracy during the test phase. Finally, we also did some experiments to understand the importance of LSTM cells in this neural network by removing it completely in one of the experiment and also replacing it with a GRU in another and concluded through these that LSTM cells allows the model to handle time step information from long interval input sequences and determine during training which information to keep and which to forget.

References:

[1] Faust et al. "Automated detection of atrial fibrillation using long shor-term memory network with RR interval signals

[2] https://physionet.org/content/afdb/1.0.0/

[3] A review on deep learning methods for ECG arrythmia classification, Zahran Ebrahimi et al.

[4] D. Clark, V. Plumb, A. Epstein and G. Kay, "Hemodynamic effects of an irregular sequence of ventricular cycle lengths during atrial fibrillation," Journal of the American College of Cardiology, vol. 30, no. 4, pp. 1039-1045, 1997.

[5] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[6] https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21?gi=77e43361ab6c

[7] https://en.wikipedia.org/wiki/Electrocardiography