UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA BIOMÉDICA

ECG SIGNAL CLASSIFIER BASED ON N-BEATS NEURAL NETWORKS

AUTHOR: Bartosz Puszkarski

SUPERVISOR: José Millet Roig

SUPERVISOR: Juan Besari

Academic year: 2019-20

Declaration of Authorship

Hereby I, *Bartosz Puszkarski* declare, that I have composed the presented thesis *ECG Signal Classifier Based on N-Beats Neural Networks* independently and have not used any printed or electronic sources other than those stated in the bibliography. All passages taken directly or indirectly from these sources have been clearly denoted as such, in each individual case with precise indication of the source.

City, Date	Name

Abstract

N-Beats neural network based classifier of heart condition using ECG input.

The automated classification of any heart condition from ECG signal is undoubtedly a challenging task to perform, not to mention the task of distinguishing even more specific subcategories when the only information available are a pure ECG record and basic characteristics of a patient. Most of the commonly used and well known classification algorithms are based either on a complex, diligently collected data which describes meticulously the domain of a given problem, or on a comparison to a previously selected archetype with a segmentation of the characteristic heart signal. The main goal of this work was to use the power of deep learning methods along with the computational benefits associated with neural basis expansion analysis for interpretable time series forecasting. The classification algorithm is based on a hypothesis that each of the signals can be described using a waveform generator which will be distinctively trained on a specific heart condition records set as these signals have unique temporal energy distribution. This idiosyncratic waveforms' behaviour is suspected to be of an essential significance. N-BEATS neural networks are being used to deliver substantial waveform generators' parameters as a results of the training process conducted on the training dataset provided by PhysioNet/Computing in Cardiology (CinC) Challenge 2020. After trained generators are obtained they generate a continuation of a given test signal trying to mimic the curve of the signals they were trained on. Aforementioned process allows to measure the distances between the original testing curve and the predicted waveforms, thus indicating the main classifying factor. The algorithm achieved the following scores using scoring methods provided by PhysioNet/CinC Challenges 2020. Accuracy at the level of 78%, AUROC: 67%, however the scores are to improve along with the training process.

Keywords: ECG, neural network, Prediction, classification, Physionet database, N-Beats

Resumen

La clasificación automática de cualquier afección cardíaca a partir de la señal de ECG es, sin duda, un desafío más si se pretende distinguir entre distintas subcategorías más específicas a partir del registro de una única derivación del electrocardiograma. El objetivo principal de este trabajo consiste en utilizar el potencial de los métodos de 'aprendizaje profundo' (Deep learning) junto con los avances en computación asociados con el "neural basis expansion analysis' para pronósticos de series de tiempo interpretables. El algoritmo de clasificación a desarrollar se basa en la hipótesis de que cada una de las señales puede describirse utilizando un generador de forma de onda.

N-BEATS se utilizarán para obtener parámetros de onda que se validarán sobre la base de datos abierta ofrecida por PhysioNet en el marco del congreso Computing in Cardiology (CinC) Challenge 2020.

Palabras clave: ECG, redes neuronales, clasificación, Physionet base de datos, N-Beats

Contents

1 Introduction and Goals **1**

2 Material and Methods **5**
2.1 Terminology clarification . . . 5
2.2 Data . . . 6
2.3 Software . . . 9
2.3.1 Language and libraries . . . 9
2.3.2 Working environment . . . 9
2.4 Data analysis . . . 10
2.4.1 General data analysis . . . 10
2.5 N-BEATS Neural Net . . . 20
2.5.1 Knowledge source . . . 20
2.5.2 Why N-BEATS? . . . 21
2.5.3 Scoring . . . 21
2.5.4 Basic Measures . . . 21
2.5.5 F-score . . . 22
2.5.6 Generalization of The Jaccard Measure . . . 23
2.5.7 N-Beats Architecture . . . 23
2.5.8 Basic Block . . . 24
2.5.9 Double Residual Method . . . 25
2.5.10 Generic Architecture . . . 25
2.6 Preprocessing . . . 26
2.7 Signal Features Extraction . . . 27
2.8 Network Inputs and Outputs . . . 27
2.9 Implementation . . . 27
2.10 Classifier . . . 35
2.10.1 How to Run . . . 35
2.10.2 Classifier Input and Output . . . 36
2.10.3 Classifier Description . . . 36

3 Results **39**
3.1 Training Results . . . 39
3.1.1 Graphs Legend . . . 39
3.1.2 Network describing LBBB ECG signal . . . 40
3.1.3 Network describing PVC ECG signal . . . 41
3.1.4 Network describing STD ECG signal . . . 43
3.1.5 Network describing AF ECG signal . . . 44
3.1.6 Network describing PAC ECG signal . . . 46
3.1.7 Network describing Normal ECG signal . . . 47
3.1.8 Network describing STE ECG signal . . . 49
3.1.9 Network describing RBBB ECG signal . . . 50
3.1.10 Network describing I-AVB ECG signal . . . 52
3.2 Classification Results . . . 53

3.3 Received Scores 53
3.3.1 Classifier trained on GPU with MSE loss function as decisive 53
3.3.2 Classifier trained on CPU with L1 loss function as decisive 56
3.3.3 Classifier trained on CPU with MSE loss function as decisive 58
3.3.4 Classifier trained on CPU with MSE loss function as decisive with weight bias on LBBB descriptor 60

4 Discussion **67**

5 Conclusion **71**

References **73**

1 Introduction and Goals

The Electrocardiogram is already claimed to be a valuable tool in any expertise associated with detection of cardiac dysfunctions. In 1891, William Bayliss and Edward Starling, British physiologists of University College London, demonstrated triphasic cardiac electrical activity in each beat using an improved capillary electrometer (Burnett, 1985). The standard 12-lead ECG has been widely used to diagnose a variety of cardiac abnormalities such as cardiac arrhythmias, and predicts cardiovascular morbidity and mortality (Kligfield et al., 2007) Yet being available for the medics for more than a hundred years now, it's potential is still surprising more and more researchers, especially when approached from the perspective of the new technology available, and the computational power connected with it - used widely in a field of Machine Learning and Deep Learning. It is now clear, that with a early and correct diagnosis of cardiac abnormalities the chances of successful treatments are increasing radically, and thus proving its usefulness (Adams et al., 2007). More than 1 million ECGs are recorded every day. However, most of the process nowadays is performed manually, requiring well trained specialists and appropriately set hardware. This implicates a wide range of errors, which might be an outcome of a human factor in the process. The findings showed that electrode placement accuracy varies from 16–90%, and standards and guidelines on electrodes placement are not being adhered to. Poor electrode placement can mean under- or overdiagnosis, which can increase morbidity and mortality, or mean that patients receive unnecessary treatment or hospitalisation. (Bickerton & Pooler, 2019). An opportunity never accessible before, now, in 21^{st} century, is shining bright in front of the people desiring to investigate it and to reach for mathematically powered tools of Data Analysis allowing them to have a powerful insight. Hardware possibilities reaching the peak of their performance discover complex new patterns never seen before and inconceivably arduous to detect by human senses or intuition. With that in mind, an approach to a further investigation of use was made and with help of Mr. Jose Millet Roig and Mr. Juan Besari an algorithm classifying heart conditions was developed, as a part of the participation process in PhysionetChallenge / Computing in Cardiology 2020. Current work describes an actual development process along with the theoretical base laying under the proposed solution and provides a mathematical background explaining the potential and justifying the chosen neural network architecture and describes the differences between this solution and the ones proposed in the past. Many attempts have been made including neural networks using 12-

lead ECG signal in order to classify cardiac dysfunctions in the past, but they mainly focused on certain parameters and features which are specific to ECG signals like the P wave which is a representative of atrial depolarisation (cardiac stimulation), the QRS complex representing ventricular depolarisation and the T wave, which represents the return of the ventricles to their resting state (re-polarisation) (Nugent et al., 1999). Another popular approach often mentioned in scientific publication from cardiac domain is a use of more advanced statistical analysis including Support Vector Machines or Gaussian Mixture Models. They base on finding ECG segmentation and calculation of log-likelihood. Also, use of the support vectors indicates, that the classifiers extract from the signal key features. (Chang et al., 2012). Chosen architecture - N-BEATS neural network, was never before used in the domain of cardiac dysfunctions classification, however it gained a lot of recognition after being published as a potentially superior to the M4 Contest winner being repeatedly mentioned among data science society. It focus on solving the univariate times series point forecasting problem using deep learning. It is a deep neural architecture based on backward and forward residual links and a very deep stack of fully-connected layers. N-Beats has a number of important features like being interpretable, applicable to a wide array of target domain. It is applicable in almost any field containing Time Series forecasting or seasonality / trend analysis. Time series forecasting underlies most aspects of modern business, including such critical areas as inventory control and customer management, as well as business planning going from production and distribution to finance and marketing (Oreshkin et al., 2019). Although it was designed with regression purposes in mind I believe that the results of this regression combined with their interpretability may prove to be adequate to perform classification on ECG signal, as it is strongly periodical and repetitive and trends for a certain dysfunctions are significantly marked. These characteristics fit tightly into the domain of time-series forecasting (Barnett et al., 2004) The difference between approach studied in this paper and others that in N-Beats, in opposition to other methods (Nugent et al., 1999) (Chang et al., 2012) (Chaitman et al., 1996) (Yu & Chou, 2008) , nothing is used to decompose ECG signals into sets of weighted basic components. Therefore it facilitates classification use and performance eliminating the need of wide analysis of the data set provided. N-Beats as a quite big neural network with tendencies to snowball it's size requires a powerful computational machine. Due to the memory requirements and the amount of computations performed, GPU use is strongly advisable. The proposed algorithm has been tested using a very recent open database available to the entire community of scientists and developers with the aim of obtaining more robust algorithms that can be part of medical devices that improve diagnosis, and even also can be used in wearable. The training data-set was provided by the

PhysionetChallenge organizers and it's additional explanation can be found below.

To highlight and clarify the objectives of this work, a goals list is written below:

1. Implement N-Beats neural networks.
2. Use of GPU-based architecture to optimize the computation required for its implementation.
3. Training neural networks to predict and describe ECG signal of distinguished classes.
4. Propose classifier system based on the previously trained neural networks which will label new ECG signal with one of the nine labels provided by the Physionet Challenge.
5. Experiment with different variations of the classifier in order to quantify the potential signal predictor of the system.

2 Material and Methods

2.1 Terminology clarification

As lots of acronyms is being used on the pages of this work along with terms specific to the problem domain, it would be advisable to explain their meaning and purpose of their use.

- AUROC - Area Under the Receiver Operating Characteristic, a measure used to evaluate multi class classification results. It tells how much model is capable of distinguishing between classes. Higher the AUROC, better the model is at distinguishing between patients with disease and no disease.
- AUPRC - Area Under the Precision-Recall Curve is another measure used to evaluate classification, this time focusing more on whether the model avoid False Positive/Negative classifications. Perfect AUPRC means that the model is able to properly classify all cases of the class, without any extensional marking other cases from different classes as this one.
- Accuracy - Due to its definition it is the ratio of number of correct predictions to the total number of input samples.
- F-measure - It is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score. Detailed explanation was made in section 2.5.5.
- Fbeta-measure - is the weighted harmonic mean of precision and recall, reaching its optimal value at 1 and its worst value at 0. Meticulously explained along F-measure in section 2.5.5.
- Gbeta-measure - Scoring function which is a generalization of the Jaccard measure, where we have given missed diagnoses twice as much weight as correct diagnoses and false alarms. It's mathematical basis and a minute explanation were presented in section 2.5.6.
- MSE Loss - Mean Squared Error function loss, is a function measuring the distance between received value and the one expected. It is calculated in the following way: $MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$.
- L1 Loss - is used to minimize the error which is the sum of the all the absolute differences between the true value and the predicted value. It is calculated using the following equation: $\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$

- WFDB - Waveform-Database. It is a file format provided by the Physionet organisation in order to provide standardized way of storing waveform signals. It is also associated with Waveform Database Library.
- MATLAB - MATLAB is a programming platform designed specifically for engineers and scientists. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.

2.2 Data

The data set for this work comes from the sources provided by PhysionetChallenge, which consists of:

1. Southeast University in China,
2. Centre for Cardio-metabolic Risk Reduction in South-Asia (CARRS) in India,
3. Record of the diverse population in the USA

However as the authors of the challenge indicate, main training set is the public data available and used during China Physiological Signal Challenge in 2018 (CPSC2018). Aforementioned training set includes 6877 standard 12-lead electrocardiogram (ECG) records out of which 53.7% of the representatives are males, and the remaining 46.3% are females (the PhysioNet/-Computing in Cardiology Challenge 2020, 2020), all sampled at the frequency of 500Hz.

Being provided in WFDB format the training sets is a combination of binary MATLAB v4 files and text files written in WFDB header format, containing electrocardiogram signal data and signal/patient description respectively, including recognised heart condition.

To depict an available input data an exemplar header file would be presented below:

```
A3674 12 500 5000 16-Mar-2020 19:07:01
A3674.mat 16+24 1000/mV 16 0 -74 -9 0 I
A3674.mat 16+24 1000/mV 16 0 -89 -33 0 II
A3674.mat 16+24 1000/mV 16 0 -15 -26 0 III
A3674.mat 16+24 1000/mV 16 0 81 12 0 aVR
A3674.mat 16+24 1000/mV 16 0 -29 -6 0 aVL
A3674.mat 16+24 1000/mV 16 0 -52 28 0 aVF
A3674.mat 16+24 1000/mV 16 0 -33 9 0 V1
A3674.mat 16+24 1000/mV 16 0 -71 -52 0 V2
A3674.mat 16+24 1000/mV 16 0 -107 -70 0 V3
```

```
A3674.mat 16+24 1000/mV 16 0 -205 -43 0 V4
A3674.mat 16+24 1000/mV 16 0 -165 -1 0 V5
A3674.mat 16+24 1000/mV 16 0 -83 -4 0 V6
#Age: 60
#Sex: Female
#Dx: STD
#Rx: Unknown
#Hx: Unknown
#Sx: Unknown
```

First row of the presented file provides information which interpretation may be found in WFDB header file description under the name of *Record Line* - being the first non-empty, non-comment line, it contains information applicable to all signals in the record. Its fields' meanings are, in order from left to right:

Table 2.1: WFDB record line attributes meaning

Index	Attribute	Description
1	record name	A string of characters identifying the record.
2	number of signals	Number of signal layers in file, must not be negative.
3	sampling frequency	Given in samples per second per signal
4	number of samples per signal	Total number of samples per signal available in a given file
5	base date	Date of recording
6	base time	Time of recording

Following 12 lines however describe each individual signal. Their format is also specific, and it's meaning is explained in the table below Table 2.2.

Table 2.2: Signal attributes description

Index	Attribute	Description
1	file name	A string of characters identifying the binary file name.
2	format + byte offset	A number of bits in memory occupied by each signal and its offset respectively
3	amplitude resolution	Given in 1000 mV units
4	ADC resolution	Resolution of analog-to-digital converter used to digitized signals.
5	ADC zero	Baseline value corresponding to 0 physical units.
6	initial value	First value of the signal
7	checksum	The checksum
8	description	Lead name

Next 6 lines provide a description of a patient and his condition Table 2.3:

Table 2.3: Patient description

Index	Attribute	Description
1	Age	Patient's age
2	Gender	Patient's gender (Male or Female)
3	Dx	Diagnosis
4	Rx	The medical prescription
5	Hx	History.
6	Sx	Symptom or surgery

Each record has one or more diagnosis' labels already assigned, chosen from the set, described below in Table 2.4:

Table 2.4: Basic diagnosis labels

Index	Abbreviation	Description
1	AF	Atrial fibrillation
2	I-AVB	First-degree atrioventricular block
3	LBBB	Left bundle branch block
4	Normal	Normal sinus rhythm
5	PAC	Premature atrial complex
6	PVC	Premature ventricular complex
7	RBBB	Right bundle branch block
8	STD	ST-segment depression
9	STE	ST-segment elevation

2.3 Software

2.3.1 Language and libraries

First of all the use of Python 3.6 as a main programming / analytical tool along with available libraries like **Numpy**, **Pandas**, **Matplotlib** and the most important **PyTorch** was specified as the key element of a newly born project idea, as aforementioned tools are one of the most popular being used in a field of Data Analysis and Machine Learning. Numpy library is a library which helps scientists all over the world to conduct mathematical calculation in fast and ordered way, facilitating complex computations and processing them in a rapid manner. Pandas is a library helping with data management and processing, especially with it's DataFrame object which allows it's users to use table alike functionalities in intuitive way. Matplotlib is a library which provides a lot of plotting utilities. It helps in both data and results visualisation, is very intuitive and well documented. PyTorch was selected due to its usefulness and documentation. It also works well on GPUs allowing to perform neural net computations in a multi threaded environment.

2.3.2 Working environment

The Project was developed on two different work stations, as it required a use of Graphical Processing Unit to prosecute computations associated with neural network training process, unavailable on one of them. Having that mentioned, both of the aforementioned machines' specifications are presented in the tables below, starting with the weaker one, which a main coding station - a laptop with a virtual environment specified clearly for the PhysionetChallenge purposes. Table 2.5

Table 2.5: Laptop characteristics

Parameter	Description
Type	Laptop
Name	Lenovo Yoga 720 13
CPU	Intel Core i7-7500U CPU @ 2.70GHz x 4
VGA	Intel HD Graphics 620 (Kaby Lake GT2)
RAM	8GB
Storage	Samsung SSD 256 GB SM961/PM961
System	Ubuntu 18.04.4 LTS 64 bit

Second, stronger machine which was used as a training unit has been lent to me as a courtesy of Juan Besari, whose help with this project was unquestionable. Training code was being sent to the computer and then run by the owner which allowed to inspect the code and proceed with minor experiments along the training process not investing any resources additionally from a weaker machine. The previous' characteristic is visible below. Table 2.6

Table 2.6: PC characteristic

Parameter	Description
Type	Personal Computer
CPU	AMD Ryzen 5 2600. Six-Core, 12 Threads. Overclocked to 4100 MHz
GPU	ZOTAC GTX 1080 8GB VRAM GDDR5X
Motherboard	X470 AORUS ULTRA GAMING
RAM	32 GB (8GB x4) 3200 MHz G.Skill Trident Z DDR4 PC4-25600 CL16
Main Storage	Samsung SSD NVME 970 EVO 500 GB
Secondary Storage	1x Kingston SA400 SSD 240 GB, 2x Western Digital HDD 3 TB 5,400 r.p.m.
System	Microsoft Windows 10 Pro (10.0.18363 compilation 18363)

2.4 Data analysis

2.4.1 General data analysis

Data analysis topic for this particular task was definitely a challenging goal to achieve, as all the measurements were conducted with only the frequency of signal reads in common.

Therefore, only **one lead** was selected to serve as a guiding signal. The selected one is

Lead III as it measures the potential between Left Arm (negative) and Left Leg (positive) electrodes providing an inferior right heart view - it served as an input to the algorithm. Why only one lead? From the beginning of this work, it was planned to use only one lead first, then after obtaining decent results to expand its number. Yet, it was challenging to achieve satisfying results with one lead, so it was decided to put all emphasis and resources to make one lead based prediction as good as it could be.

As the classification task of the 9 aforementioned classes is undoubtedly arduous it might be worth mentioning how does the data propagation looks like, when talking about the given signals. Because of the classification performed it is crucial to know how does each of the signals type look like, in order to eventually interpret the outcomes. Because of that, an addition description has been provided in each section, giving more in-depth look into how the classes are divided. The main goal of this section is to explain to the Reader what kind of signals are provided to the algorithm and how should behave an outcome curve - our result for each individual N-Beats module. Therefore the plots of each diagnosis type available in the training data set (basic diagnosis) are depicted below as visible by the neural net:

2.4.1.1 Atrial Fibrillation

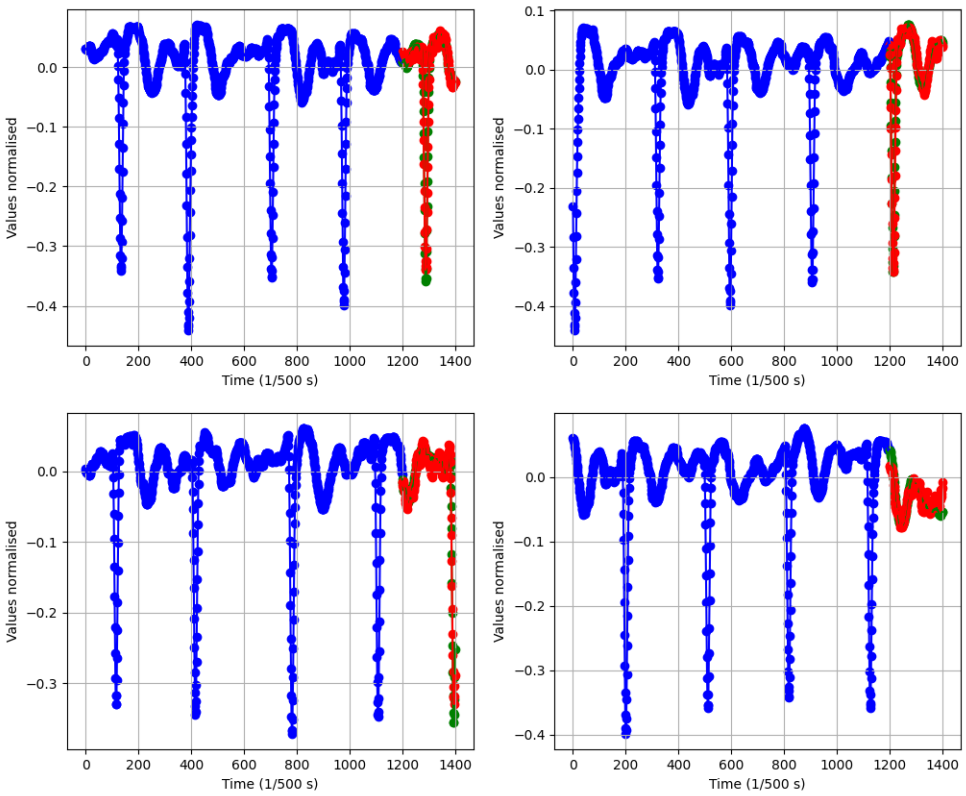

Figure 2.1: The sheer AF diagnosis signal

Abnormal heart rhythm characterized by irregular and rapid beating of the atrial chambers of the human heart. Often begins as short periods of abnormal beating, which became longer or continuous over time. During atrial fibrillation, the heart's two upper chambers (the atria) beat chaotically and irregularly — out of coordination with the two lower chambers (the ventricles) of the heart. Atrial fibrillation symptoms often include heart palpitations, shortness of breath and weakness. High blood pressure and valvular heart disease are the most common alterable risk factors for AF (Anumonwo & Kalifa, 2014) (Bun et al., 2015). Diagnostic criteria include:

1. P waves are absent.
2. There are fibrillation (f) waves instead of P waves. The f waves result in an oscillating irregular baseline.
3. he R-R intervals are not equal resulting in an irregular rhythm.

2.4.1.2 First-degree atrioventricular block

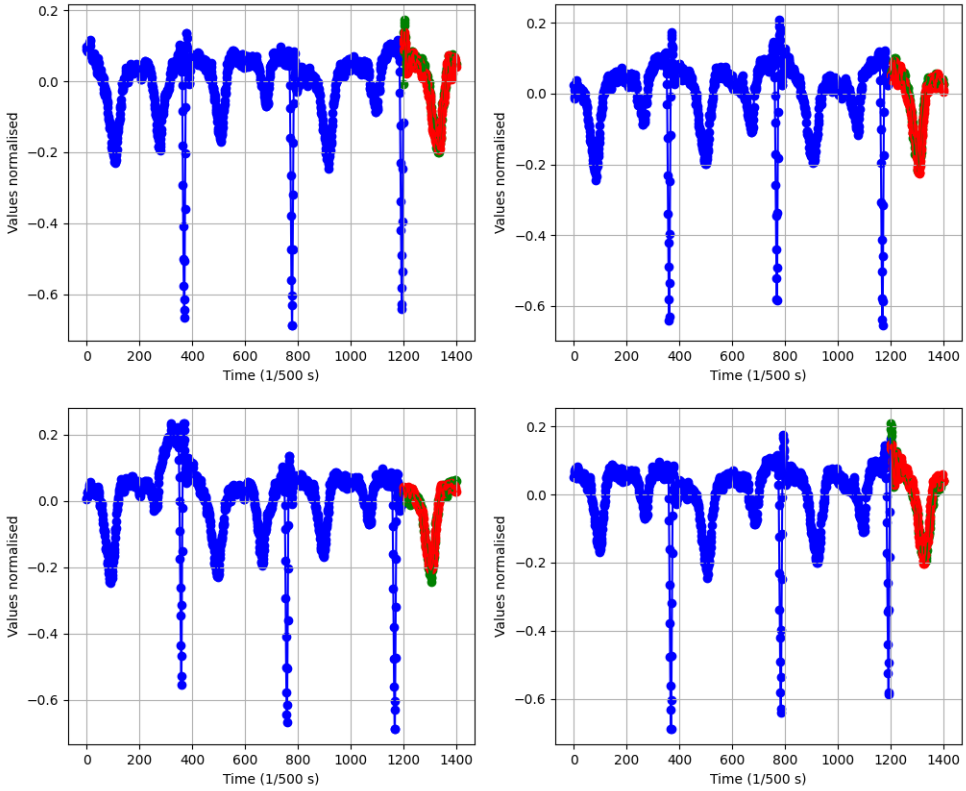

Figure 2.2: The sheer I-AVB diagnosis signal

First-degree atrioventricular block (AV block) is a disease of the electrical conduction system of the heart in which electrical impulses conduct from the cardiac atria to the ventricles through the atrioventricular node (AV node) more slowly than normal. First degree AV block not generally cause any symptoms, but may progress to more severe forms of heart block such as second- and third-degree atrioventricular block. Diagnostic criteria include:

1. PR interval is prolonged: >200 ms.
2. Every P wave is followed by a QRS complex.
3. PR interval is fixed.
4. Block is at the level of AV node.
5. Sometimes, PR interval prolongation may coexist with 2:1 second degree AV block.

2.4.1.3 Left Bundle Branch Block

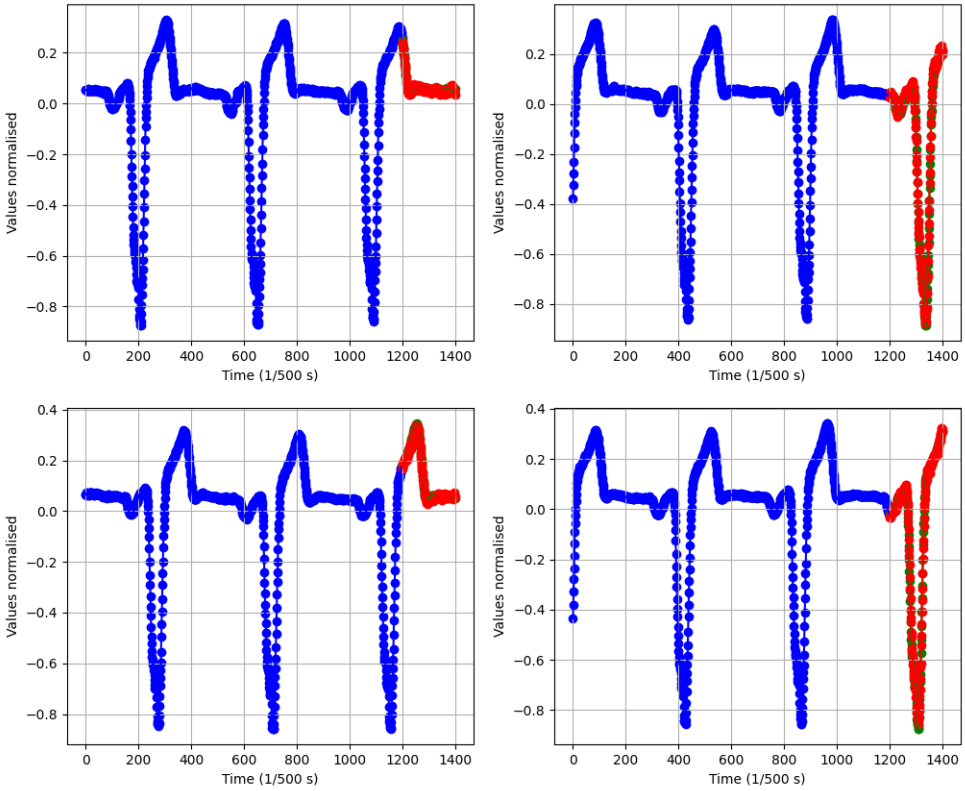

Figure 2.3: The sheer LBBB diagnosis signal

Left bundle branch block (LBBB) is a condition, in which activation of the left ventricle of the heart is delayed, which causes the left ventricle to contract later than the right ventricle. Diagnostic criteria include:

1. The QRS width should be greater than or equal to 120 ms for adults.
2. Wide, notched R wave in leads I, aVL, V5 and V6. Occasionally, RS pattern may be seen in leads V5 and V6.
3. Q waves are absent in leads I, V5 and V6. A small q wave may be present in lead aVL.
4. ST and T wave are generally opposite in direction to QRS.
5. R peak time is >60ms in leads V5 and V6. If there are small r waves in leads V1, V2 or V3, the R peak time may be normal in these leads.

2.4.1.4 Normal sinus ECG signal

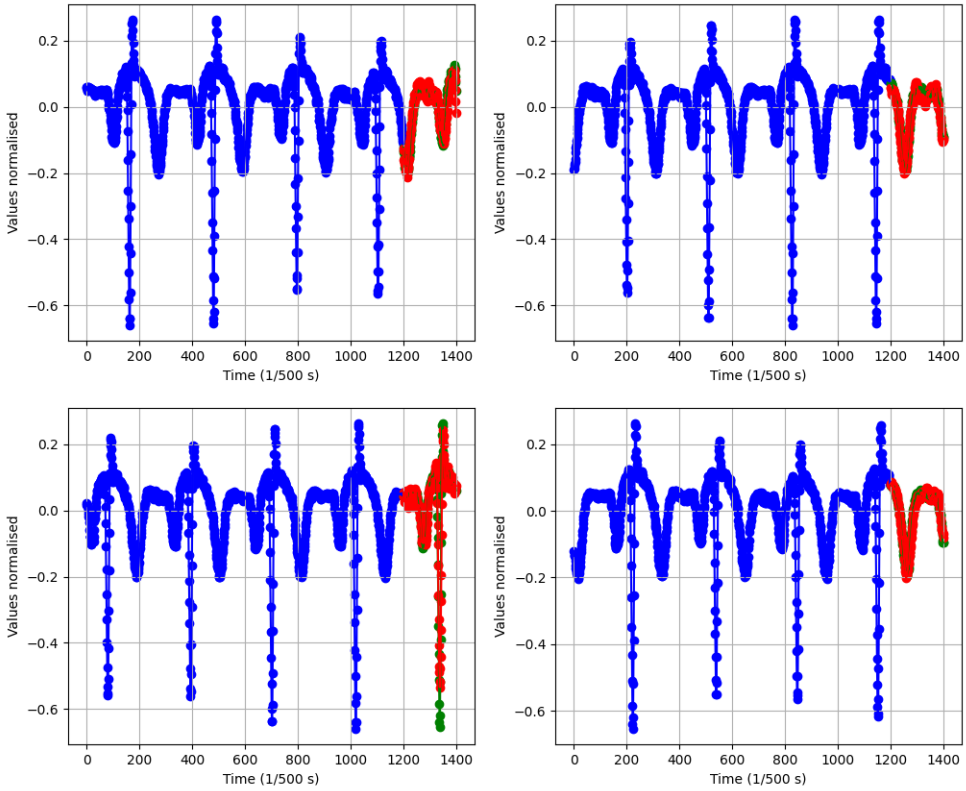

Figure 2.4: The Normal heart beat ECG signal

Normal ECG read, without any disorders. It is rhythmic, contains a strong sinusoidal trend, intervals are fixed and repetitive.

2.4.1.5 Premature Atrial Complex

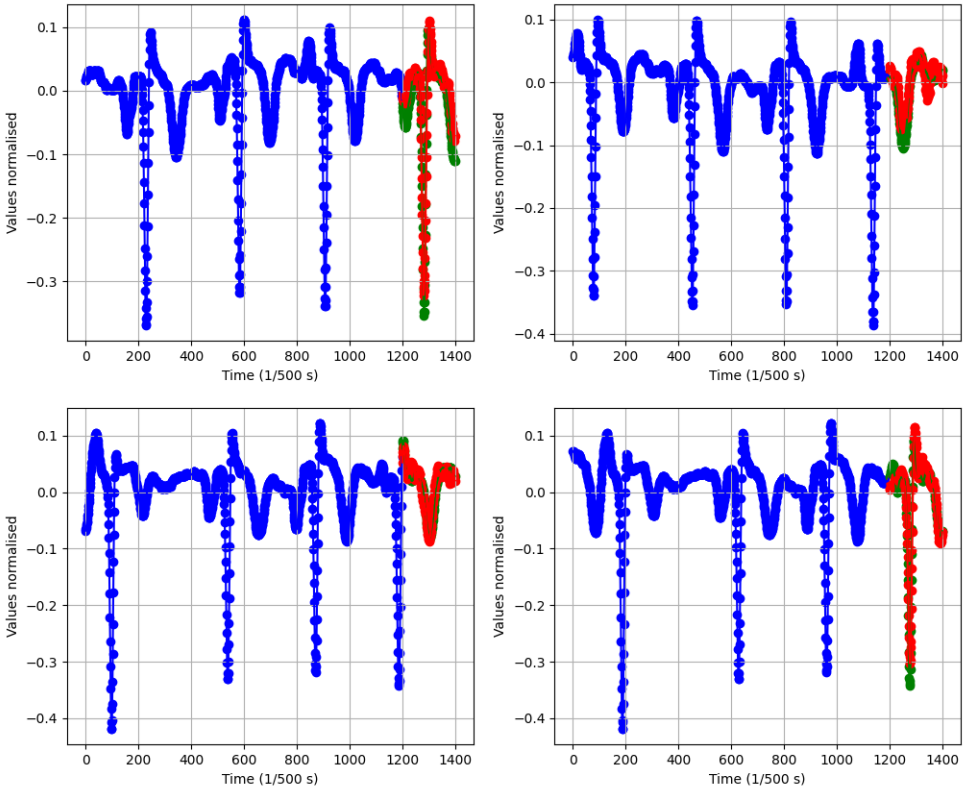

Figure 2.5: The sheer PAC diagnosis signal

Premature Atrial Complex (PAC) is a common cardiac arrhythmia characterized by premature heartbeats originating in the atria. While the sinoatrial node typically regulates the heartbeat during normal sinus rhythm, PACs occur when another region of the atria depolarizes before the sinoatrial node and thus triggers a premature heartbeat. (Cleland, 1991) Diagnostic criteria include (Alper et al., 2013):

1. The impulse arises from an atrial focus, not from the sinus node.
2. The premature atrial beat is often conducted normally to the ventricles, creating a narrow QRS complex.
3. The PR interval of the APC may be shorter or longer when compared with the PR interval of the sinus beat. If ectopic atrial focus is closer to the AV node, then ectopic impulse will reach the AV node in a shorter time.

2.4.1.6 Premature Ventricular Contraction

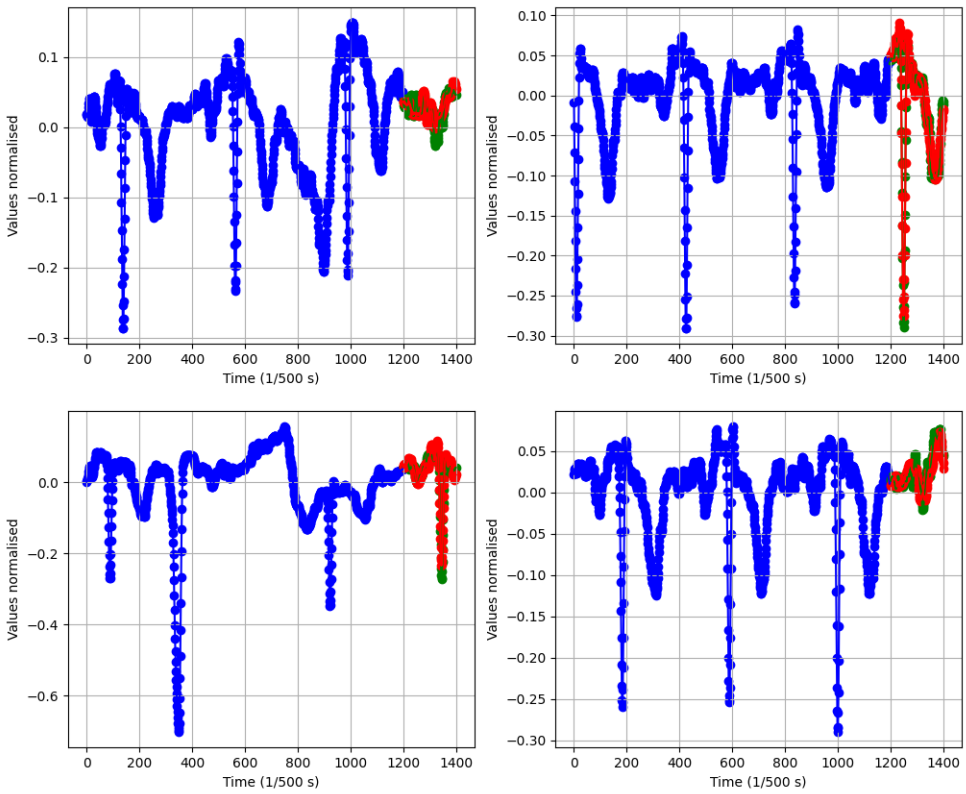

Figure 2.6: The sheer PVC diagnosis signal

Premature Ventricular Contraction (PVC) is a relatively common event where the heartbeat is initiated by Purkinje fibers in the ventricles rather than by the sinoatrial node. PVCs may cause no symptoms or may be perceived as a "skipped beat" or felt as palpitations in the chest. Single beat PVCs do not usually pose a danger. (Gerstenfeld & Marco, 2019) Diagnostic criteria include:

1. Broad QRS width being higher than 120 ms with abnormal morphology.
2. Premature — i.e. occurs earlier than would be expected for the next sinus impulse.
3. Discordant ST segment and T wave changes.
4. Usually followed by a full compensatory pause.
5. Retrograde capture of the atria may or may not occur.

2.4.1.7 Right Bundle Branch Block

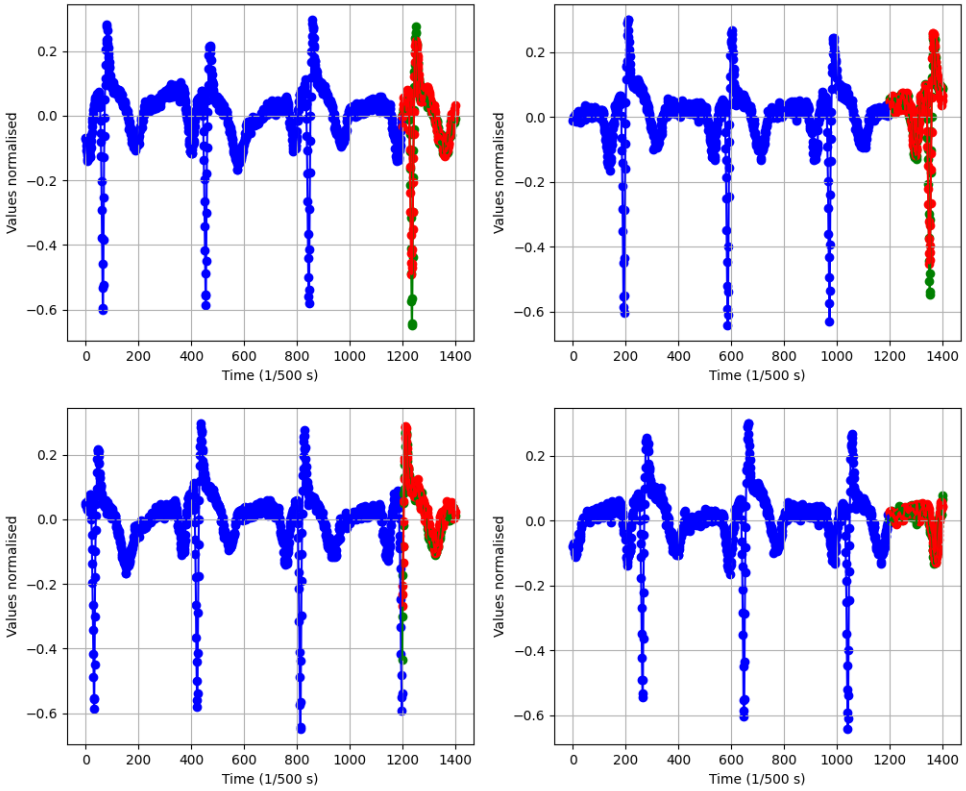

Figure 2.7: The sheer RBBB diagnosis signal

Right Bundle Branch Block (RBBB) is a heart block in the right bundle branch of the electrical conduction system. Diagnostic criteria include (Gerstenfeld & Marco, 2019):

1. The QRS width is more than 120 ms for adults
2. Premature — i.e. occurs earlier than would be expected for the next sinus impulse.
3. Discordant ST segment and T wave changes.
4. Usually followed by a full compensatory pause.
5. Retrograde capture of the atria may or may not occur.

2.4.1.8 ST Segment Depression

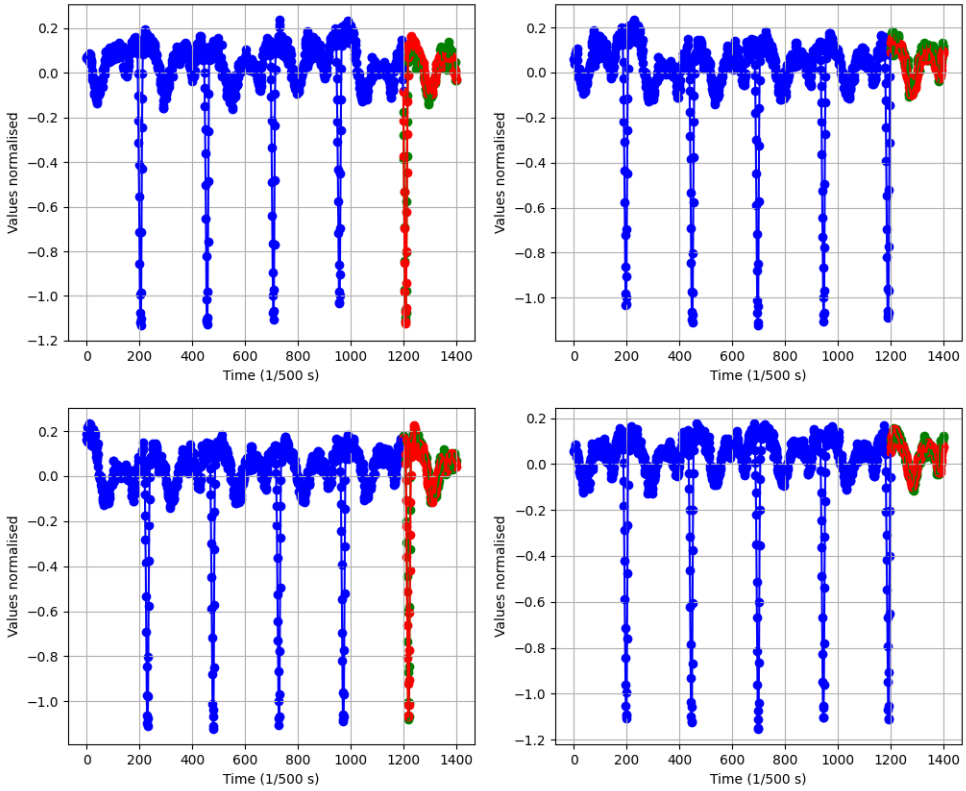

Figure 2.8: The sheer STD diagnosis signal

ST Segment Depression (STD) is characterized by a horizontal or downsloping ST segment. In many instances is associated with acute coronary syndromes — both acute ischaemia and acute infarction. (Pollehn et al., 2002) The transition from ST segment to T-wave is more abrupt in ischemia (the transition is normally smooth)

2.4.1.9 ST Segment Elevation

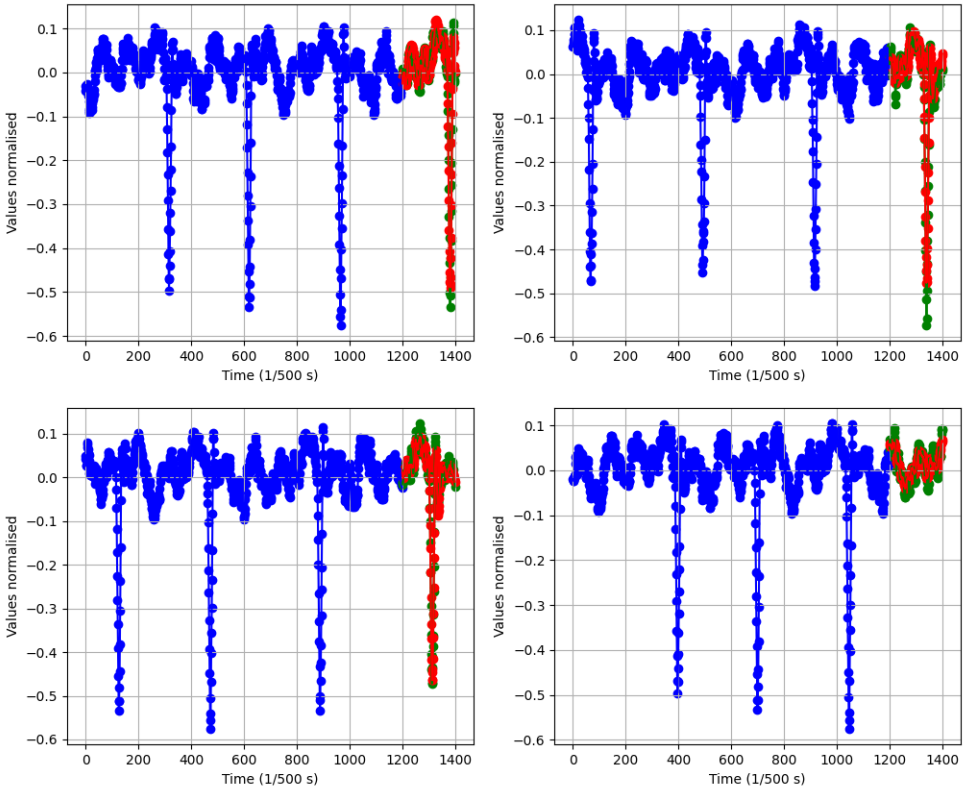

Figure 2.9: The sheer STE diagnosis signal

ST Segment Elevation (STE) refers to a finding on an electrocardiogram wherein the trace in the ST segment is abnormally high above the baseline. Often associated with myocardial infraction

2.5 N-BEATS Neural Net

2.5.1 Knowledge source

As this work is of experimental nature it has to be proclaimed, that the N-BEATS architecture, used in the project, was based on an article: *N-BEATS: Neural basis expansion analysis for interpretable time series forecasting* (Oreshkin et al., 2019) being an experimental use case for aforementioned theory. It was a base for experiences associated with the implementation, exploring the problem space and a forming factor for the way of understanding the cardiac signal classification problem.

2.5.2 Why N-BEATS?

The N-BEATS network authors' reference empirical comparison results of their implementation's efficiency to the one achieved by the winners of **The M4-Competition** claiming, that their pure Machine Learning based solution reached significantly better outcomes set side by side with standard and well-known pure statistical methods (Makridakis et al., 2018). Furthermore, relying on the achieved results being about 3% better than the ones obtained by the winner of the competition in 2018 they prove their approach being undoubtedly worth further investigation of potential uses in the area of interpretative time series forecasting. What is more, aside of the benefits associated with the accuracy of the generic N-BEATS outputs, the net might be also use to receive an interpretative results of seasonality and trend (Oreshkin et al., 2019).

2.5.3 Scoring

As the **PhysionetChallenge** rules implied, a specific scoring measures are taken into consideration to evaluate the quality of a given classifier. Implementation of these were however provided by organisers of the challenge, thus letting the participants to focus more on the classifiers implementation. Therefore it would be advisable to present a mathematical base of the first as it would allow the reader to comprehensively understand under what criterion were the final models selected.

2.5.4 Basic Measures

An arguably complete view of a systems performance is given by the precision-recall curve. Having the precision and recall mentioned a illustrative description allowing for a better understanding might be useful and thus is presented below.

For presentation purposes a following simple setting is considered: having an object with a binary label l corresponding to correct object association with respect to the task at hand, the system produces an assignment label z indicating whether it believes the object to be correct or not. Such an outcome might be summarised in a confusion table (C. & E., 2005) Figure 2.10:

		Assignment z	
		+	-
Label ℓ	+	TP	FN
	-	FP	TN

Figure 2.10: Confusion table (C. & E., 2005)

Where + and − are the binary classification classes. TP stands for true positive, respectively TN stands for true negative and FP means false positive, FN is false negative. Having those in mind it is possible to compute precision (p) and recall (r).

$$p = \frac{TP}{TP + FP} \tag{2.1}$$

$$r = \frac{TP}{TP + FN} \tag{2.2}$$

2.5.5 F-score

F-score is a weighted harmonic average of precision and recall (C. & E., 2005), calculated as follows:

$$F_\beta^l = (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision \cdot recall} = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + FP + \beta^2 \cdot FN}, \beta = 2 \tag{2.3}$$

Being calculated over all recordings, this F1-score is weighted by the associated importance of diagnosis, following the equation:

$$F_\beta = \frac{1}{N}\sum_{t=1}^{N} C^l \cdot F_\beta^l \tag{2.4}$$

where:

- C^l is the importance of class l.
- N is the number of classes.
- $C^l = 1$ initially, invoking that each class is equally important.

2.5.6 Generalization of The Jaccard Measure

The Jaccard index is a classical similarity measure on sets with a lot of practical applications in information retrieval, data mining, machine learning and many more. Measuring the relative size of the overlap of two finite sets A and B, the Jaccard index J is formally defined as (Kosub, 2019):

$$J(A,B) =_{def} \frac{|A \cap B|}{|A \cup B|} \tag{2.5}$$

and the associated Jaccard distance J_δ is known to fulfill all properties of a metric, notably, the triangle inequality. It is formally described as (Kosub, 2019):

$$J_\delta(A,B) =_{def} 1 - J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} = \frac{|A \triangle B|}{|A \cup B|} \tag{2.6}$$

Yet, in case of the PhysionetChallenge scoring method, a generalised approach was used, basing on the previously presented measures of precision and recall, giving the missed diagnosis twice as much weight, as the correct one. Thus, for each class the following equation is computed:

$$G^l_\beta = \frac{TP}{TP + FP + \beta \cdot FN}, \beta = 2 \tag{2.7}$$

Analogically to the aforementioned F1-score, also this one is computed over all recordings:

$$G_\beta = \frac{1}{N} \sum_{t=1}^{N} C^l \cdot G^l_\beta \tag{2.8}$$

It is also indicated, that as the recordings of certain cases have multiple labels, contributions to these scoring functions are normalized, so the recordings, not class themselves provide equal contribution.

2.5.7 N-Beats Architecture

N-Beats neural network architecture design methodology was based on a handful of key principles. First, it's base architecture should be simple, clean and generic, yet deep and expressive at the same time. Second, the architecture should nor rely on time-series-specific feature engineering or input data scaling. The architecture itself should be extendable towards making its outputs interpretable.

The basic building block proposed by the article authors consists of a fork architecture and is

presented in Figure 2.11. It is also built from a multi-layer FC network (fully connected) with ReLU non-linearity. Predicting the basis expansion coefficient θ_f (forecast) and backward θ_b backcast, basic block is the vital point for the properly functioning algorithm. These blocks are organized into stacks using doubly residual stacking principle. Stack's layers may share g_θ.Building a very deep neural networks with interpretable outputs is an outcome of the forecast being aggregated in a hierarchical fashion (Oreshkin et al., 2019).

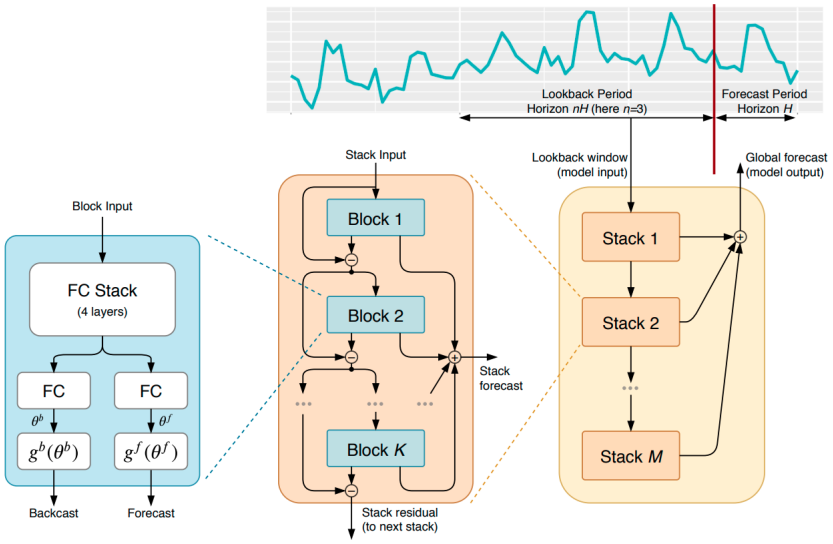

Figure 2.11: N-Beats proposed architecture.

2.5.8 Basic Block

Having a fork-alike structure, basic block is depicted on the Figure 2.11 as the left-most one, it accepts an input x and outputs two vectors, $\hat{x}$ and $\hat{y}$. Input x is a certain history lookback window of a certain length ending with the last measured observation, $\hat{y}$ is the block;s forward forecast of length H, and $\hat{x}$ is the block's best estimate of x, also known as the 'backcast'. Backcast estimation is performed given the constraints on the functional space, that the block can be used to approximate signals. (Oreshkin et al., 2019). In the architecture chosen to develop the classifier, the lookback window length is set to $5H$ which means, that it is of length equal to forecast horizon multiplied by 5. It is a firm statement, as due to the authors of the N-Beats neural net typical lookback window lengths are $x \in < 2H; 7H >$.
Inside, the basic block consists of two parts parts, first of which is a waveform generator $g_\theta : \tau^N \rightarrow \gamma^N$ being responsible for mapping N points from the domain of time

$\tau^N \subset R^N$ to N points in a domain of forecast / backcast values $\gamma^N \subset R^N$. The number of forecast points and the number of backcast points are different in the general case. Waveform generator thus is tuned with $\theta \in \Theta \subset R^M$ Function of g_θ is to provide sufficiently prosperous set of time-varying wave-forms, selected accordingly to the θ parameter variation. Aforemen-

tioned g_θ parameter might be either a component part of the learning process obtained as a part of its' results or might be limited in an advance due to the provided restrictions and constraints of the structure of outputs.
Second part makes a forecast and backcast predictors of wave-forms generators' parameters. It's fork-alike architecture might be described using the following equation:

$$\begin{aligned} \varphi_\phi^f &: R^{dim(x)} \to \Theta \\ \varphi_\phi^b &: R^{dim(x)} \to \Theta \end{aligned} \tag{2.9}$$

The tasks of φ_ϕ^f include the prediction process of future θ^F parameter expansion, which, in a final result, will lead to further optimization of the partial prediction $\hat{y}$. Analogically, the tasks of φ_ϕ^b include providing estimations of $x$with the final goal to aid in the input data description optimisation by having it selected and partially reduced (especially the parts which are not helpful anymore in the prediction process) before passing it to the blocks laid below (Oreshkin et al., 2019).

2.5.9 Double Residual Method

The middle block of the N-Beats architecture depicted on the figure Figure 2.11 consists of two residual branches, one of which conducts the backcast prediction on each of it's layers, and the second proceeds with the proper prediction for the previously mentioned future horizon on each of it's layers. The branch performing the backcast residual can be described as the one running a sequential analysis on the given input data. Each of the blocks removes a part of the input signal which it is able to describe well, thus resulting in a forecast prediction process being easier along the downstream blocks. What is more, such a solutions allows for a facilitation of more fluid gradient back-propagation. A partial forecast predictions are the outputs of each layer providing a hierarchical decomposition, being first aggregated at the stack level and then at the overall network level. Were the stacks allowed to have an arbitrary g_θ for each layer and the architecture model was not generic, it would have a different effect on the network, other that simply making the network more transparent to gradient flows.

2.5.10 Generic Architecture

Two different approaches/ architectures can be distinguished in N-Beats neural network architecture due to it's creators - a generic or interpretable. The aforementioned architectures names are associated with the returned values structures and their meaning. For the generic network architecture this means that the network is returning pure values of it's predictions. For the interpretable neural network its' outputs have a direct translation to the seasonality

and the trend line as a functions.

For the sake of this work, generic architecture is being favoured, as the one giving more possibilities with the numeric operations, which were a base of the proposed classifier. It does not rely on any knowledge or assumptions specific for problems from the field of the time-series forecasting.

Parameter g_θ is being set to be a linear projection of the previously obtained value on the previous output layer. In such a case, a partial prediction (forecast) exiting block j in i stack is being defined as follows(Oreshkin et al., 2019):

$$\hat{y}_{i,j} = \mathbf{W}_{i,j}\theta_{i,j} + \mathbf{b}_{i,j} \tag{2.10}$$

Generic model may be described and explained more visually to allow readers not being heavily involved in machine learning studies understand the topic better: FC layers embedded in the basic building block of the N-Beats network presented at 2.11 learn to predict decomposition of the partial forecast $\hat{y}_{i,j}$ of the $\mathbf{W}_{i,j}$ base already learned by the neural network. Matrix $\mathbf{W}_{i,j}$ is of $dim(\theta_{i,j})$ dimensions, implying, that the first dimension of $\mathbf{W}_{i,j}$ may be interpreted as a discrete time index in the forecast domain. Additionally, the second dimension of the aforementioned matrix might be interpreted as the basic functions' parameters, whose expansion coefficient is $\theta_{i,j}$ (Oreshkin et al., 2019)

2.6 Preprocessing

A fair amount of preprocessing is being performed in order to prepare data for the further process of classification. Firstly, only the Lead III recording signal and corresponding header is taken from the data file, it is being flattened and converted into 1-D Numpy array to facilitate further calculations. Then "min-max" normalisation is being performed, which means that the resultative dataset fulfill the following equation:

$$z = \frac{x - min(x)}{max(x) - min(x)} \tag{2.11}$$

Thanks to that, all values which are later studied by the neural network are falling in range between 0 and 1. After performing normalisation, data is split into batches, training set of which is of $((provided\_signal\_length - forecast\_length) x backcast\_length)$ dimension. Testing set for predictions is created in an analogical manner and has dimension of $((provided\_signal\_length - forecast\_length) x forecast\_length)$ Later, the amount of recordings in a single read file is being checked, and if it exceeds the range of 7500 reads

(which is equal to 15s of recording) and if so, only first 15 seconds are taken into consideration in order to avoid one file overtaking over the data-set due to its size. Batches associated with the later recording in this file are simply abandoned.

2.7 Signal Features Extraction

No feature selection is being performed in this method. The idea was to use sheer time series forecasting approach in order to get the neural network available to describe and predict future behaviour of actually investigated signal only by providing the signal itself, without any additional pre-processing performed to extract time-series forecasting specific features , making it an universal ECG descriptive tool. Time series forecasting tools proved to be applicable in the domain of ECG signals analysis (Barnett et al., 2004) and thus N-Beats neural net seemed to a perfect candidate as it doesn't require any specific time-series features declaration (Oreshkin et al., 2019). The underlying goal of this classifier was to achieve results without feature selection comparable with the already well-known ones obtained from the classifiers which depend heavily on the future selection and analysis.

2.8 Network Inputs and Outputs

Algorithm input is a single ECG read binary file provided in WFDB format as explained in the first subsection of this chapter, however to the network itself a single input vector is being passed. The input vector x is an one dimensional array, with floating-point numerical values from the third lead from 12-lead ECG signal. It has a set length of $backcast\_length$ attribute which corresponds to the length of the lookback window as indicated in N-Beats documentation (Oreshkin et al., 2019). The length of the lookback window was found during the excessive process of the parameters' grid search and equals $backcast_length = 1500$ for calculations conducted on GPU, and $backcast_length = 1200$ for the ones performed on CPU. Output obtained from the network is a one-dimensional vector $\hat{y}$ of floating-point numbers which is a prediction based on the input vector. It is of length $forecast_length = 500$ for calculations performed on GPU and $forecast_length = 200$ - the values were found during extensive parameters' grid search process.

2.9 Implementation

Implementation of the theoretical background given by the document aforementioned above, is an undoubtedly challenging task, which would take tremendous amounts of time, especially given the lack of proficiency in PyTorch / Keras / TensorFlow library. Therefore it was

decided to firstly for a solution available online. One of such was found, openly available on GitHub under the MIT License ("N-BEATS: Neural basis expansion analysis for interpretable time series forecasting Implementation in Pytorch by @philipperemy (Philippe Remy)", n.d.). Thus, the implementation used to implement the N-Beats neural network in this project was inspired by the implementation done by Philippe Remy ("N-BEATS: Neural basis expansion analysis for interpretable time series forecasting Implementation in Pytorch by @philipperemy (Philippe Remy)", n.d.).
One of the crucial part of the neural net is naturally the basic building block and so is it's implementation. After describing the theoretical part it would be advisable to visualise it's use and implementation, which is done below:

Listing 2.1: Basic Block class implementation

```
class Block(nn.Module):


def __init__(self, units, thetas_dim, device, backcast_length=10,
    ↪ forecast_length=5, share_thetas=False):
    super(Block, self).__init__()
    self.units = units
    self.thetas_dim = thetas_dim
    self.backcast_length = backcast_length
    self.forecast_length = forecast_length
    self.share_thetas = share_thetas
    self.fc1 = nn.Linear(backcast_length, units)
    self.fc2 = nn.Linear(units, units)
    self.fc3 = nn.Linear(units, units)
    self.fc4 = nn.Linear(units, units)
    self.device = device
    self.backcast_linspace, self.forecast_linspace = linspace(
        ↪ backcast_length, forecast_length)
    if share_thetas:
        self.theta_f_fc = self.theta_b_fc = nn.Linear(units, thetas_dim)
    else:
        self.theta_b_fc = nn.Linear(units, thetas_dim)
        self.theta_f_fc = nn.Linear(units, thetas_dim)


def forward(self, x):
```

```
        x = F.relu(self.fc1(x.to(self.device)))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = F.relu(self.fc4(x))
        return x

    def __str__(self):
        block_type = type(self).__name__
        return f'{block_type}(units={self.units},␣thetas_dim={self.thetas_dim
            ↪ },␣' \
               f'backcast_length={self.backcast_length},␣forecast_length={self
                   ↪ .forecast_length},␣' \
               f'share_thetas={self.share_thetas})␣at␣@{id(self)}'
```

As presented at Listing 2.1, four Full Connected layers are the entering point's of the block, being followed by either on or two additional Full Connected layers which are the representations of the thetas parameters estimators both for backcast and forecast. A decisive factor, saying whether a two separate Full Connected layers are created or just a single, shared one, is the parameter *share_thetas*.
ReLU activation functions are called in an overridden function *forward* passing the x input vector to the next layers consecutively.

As the Generic architecture was chosen as the destined ones for the purposes of ECG signal mimicking, more detailed class of the Generic Block should be considered as an obligatory to present one, and so is done:

Listing 2.2: Generic Block implementation

```
class GenericBlock(Block):

    def __init__(self, units, thetas_dim, device, backcast_length=10,
        ↪ forecast_length=5):
        super(GenericBlock, self).__init__(units, thetas_dim, device,
            ↪ backcast_length, forecast_length)

        self.backcast_fc = nn.Linear(thetas_dim, backcast_length)
```

```
        self.forecast_fc = nn.Linear(thetas_dim, forecast_length)

    def forward(self, x):
        # no constraint for generic arch.
        x = super(GenericBlock, self).forward(x)

        theta_b = F.relu(self.theta_b_fc(x))
        theta_f = F.relu(self.theta_f_fc(x))

        backcast = self.backcast_fc(theta_b) # generic. 3.3.
        forecast = self.forecast_fc(theta_f) # generic. 3.3.

        return backcast, forecast
```

The point of the class visible in Listing 2.2 is to create an object specifically designed to be a part of a generic architecture, setting more general parameters, previously not set for a general case scenario in a *Block* class, as an example backcast / forecast parameters fully connected layers might be mentioned as they are specifically initialised with respect to values of *thetas_dim* parameter along with the lengths of forecast and backcast.

Having discussed the blocks structure and goals, it is a good time to bring a doubly residual stack implementation up to the light.

Listing 2.3: Doubly Residual Stack creation

```
    def create_stack(self, stack_id):
        stack_type = self.stack_types[stack_id]
        print(f'|␣--␣␣Stack␣{stack_type.title()}␣(#{stack_id})␣(
            ↪ share_weights_in_stack={self.share_weights_in_stack})')
        blocks = []
        for block_id in range(self.nb_blocks_per_stack):
            block_init = NBeatsNet.select_block(stack_type)
            if self.share_weights_in_stack and block_id != 0:
                block = blocks[-1] # pick up the last one when we share
                    ↪ weights.
            else:
                block = block_init(self.hidden_layer_units, self.thetas_dim[
```

```
                ↪ stack_id],
                                self.device, self.backcast_length, self.
                                    ↪ forecast_length)
            self.parameters.extend(block.parameters())
        print(f'␣␣␣␣␣|␣--␣{block}')
        blocks.append(block)
    return blocks
```

Method visible above in Listing 2.3 create an array of blocks of a given type, which in the case of this project is a generic block type, with also paying attention to the parameter of shared weights. The quantity of the blocks in a given stack is established by the parameter *nb_blocks_per_stack* passed while initialising the class. With respect to that and the type of the blocks set at the beginning, the resultant doubly residual stack is ready and functional to optimise gradient flow during the learning stage.

Listing 2.4: Doubly Residual Stack forward method implementation

```
def forward(self, backcast):
    forecast = torch.zeros(size=(backcast.size()[0], self.forecast_length
        ↪ ,))
    for stack_id in range(len(self.stacks)):
        for block_id in range(len(self.stacks[stack_id])):
            b, f = self.stacks[stack_id][block_id](backcast)
            backcast = backcast.to(self.device) - b
            forecast = forecast.to(self.device) + f
    return backcast, forecast
```

in Listing 2.4 a flow between the blocks is implemented for both backcast and forecast predictors. Also, using *.to(device)* pushes objects to the *CUDA* device, on which the calculation are being performed. Also, the first forecast results are being initialised at the beginning and then modified along the gradient flow process.

Also, the training loop code is worth being inspected more meticulously, given the amount of date accessible and the hardware limitation faced. For each diagnosis class the same procedure is being repeated. Should one investigate a batching process for a specific data

file, function *one_file_training_data* is the exact place to look. It reads one file from the directory passed to the function, in this place an assumption is made, that all files in this directory have the same diagnosis, being the training set for a N-Beats network associated with a specific class, thus indicating the similarity of the signals.

After opening the file containing 12 standard lead ECG signal and the diagnosis description, a decision was made to use only Lead 3 as a representative. Decision was supported by the claim that Lead III is classified as one of the best at identifying STEMIs (ST Elevation Myocardial Infarction). This allows the program to omit the rest of the data given in a file, making the hardware limitations more bearable. Signals are then flattened and are being batched - parted into smaller chunks of a set size (batches). These batches quantity is then checked, aiming to prevent running out of the memory. If the amount is extending its limitation then only a quarter of it is used in the calculation process. Finally, the received batches' array is divided into training and testing data sets using already implemented and widely recognized function *train_test_split* available in *SciKit* library.

Listing 2.5: Loading the data from a single file

```
def one_file_training_data(data_dir, file, forecast_length, backcast_length,
    ↪  batch_size):
    normal_signal_data = []
    normal_signal_x = []

    x = wfdb.io.rdsamp(data_dir + file[:-4])
    normal_signal_data.append(x[0][:, 3])
    normal_signal_x.append(range(0, int(x[1]['sig_len'])))

    normal_signal_data = [y for sublist in normal_signal_data for y in
        ↪ sublist]
    normal_signal_x = [y for sublist in normal_signal_x for y in sublist]
    normal_signal_data = np.array(normal_signal_data)
    normal_signal_x = np.array(normal_signal_x)
    normal_signal_data.flatten()
    normal_signal_x.flatten()

    norm_constant = np.max(normal_signal_data)
    normal_signal_data = normal_signal_data / norm_constant
```

```
    x_train_batch, y = [], []
    for i in range(backcast_length, len(normal_signal_data) - forecast_length
        ↪ ):
        x_train_batch.append(normal_signal_data[i - backcast_length:i])
        y.append(normal_signal_data[i:i + forecast_length])

    x_train_batch = np.array(x_train_batch) # [..., 0]
    y = np.array(y) # [..., 0]

    if len(x_train_batch) > 30000:
        x_train_batch = x_train_batch[0:int(len(x_train_batch) / 4)]
        y = y[0:int(len(y) / 4)]

    c = int(len(x_train_batch) * 0.8)
    x_train, x_test, y_train, y_test = train_test_split(x_train_batch, y,
        ↪ test_size=0.005, random_state=17)
    data = data_generator(x_train, y_train, batch_size)

    return data,x_train, y_train, x_test, y_test, norm_constant

data, x_train, y_train, x_test, y_test, norm_constant = naf.
    ↪ one_file_training_data(actual_class_dir, file, forecast_length,
    ↪ backcast_length, batch_size)
```

After obtaining a single file training data, running the training loop is being performed. A *while* loop is being performed as long as the training process doesn't stuck at certain level, resulting in only minimal *evaluation_score* change at the level of 0.00001 or an epoch limit is met (being set to 100 epochs).

Listing 2.6: Training loop

```
while difference > threshold and i < limit :
    i += 1
    global_step = naf.train_full_grad_steps(data, device, net, optimiser,
        ↪ test_losses, training_checkpoint, x_train.shape[0])
    new_eval = naf.evaluate_training(backcast_length, forecast_length, net,
        ↪ norm_constant, test_losses, x_test, y_test, the_lowest_error,
```

```
    ↪ device)
print(f"GlobalStep: {global_step}, New evaluation sccore: {new_eval}")
if new_eval < old_eval:
    difference = old_eval - new_eval
    old_eval = new_eval
    with torch.no_grad():
        print("New evaluation value:", new_eval, "  iteration:", i)
        print("Saving...")
        new_checkpoint_name = str(checkpoint_name[:-3]+str(len(
            ↪ test_losses))+".th")
        naf.save(new_checkpoint_name, net, optimiser, global_step )
```

In Listing 2.6 *Evaluation* function is being called to perform evaluation of the most actual neural network weights - it is done by performing *MSE_Loss* function from PyTorch library on the whole testing data set. As the *Evaluation* function was mentioned, more in-depth look might be advisable to understand the mechanism behind it's name.

Listing 2.7: Function performing evaluation

```
def evaluate_training(backcast_length, forecast_length, net,
    ↪ norm_constant, test_losses, x_test, y_test, the_lowest_error,
    ↪ device, plot_eval=False):
net.eval()
_, forecast = net(torch.tensor(x_test, dtype=torch.float))

if device.type == 'cuda':
    singular_loss = F.mse_loss(forecast, torch.tensor(y_test, dtype=torch
        ↪ .float)).item()
else :
    singular_loss = F.mse_loss(forecast, torch.tensor(y_test, dtype=torch
        ↪ .float).to(device)).item()

test_losses.append(singular_loss)
if singular_loss < the_lowest_error[-1]:
        the_lowest_error.append(singular_loss)
p = forecast.detach().cpu().numpy()
```

```
    if plot_eval:
        subplots = [221, 222, 223, 224]
        plt.figure(1)
        for plot_id, i in enumerate(np.random.choice(range(len(x_test)), size
            ↪ =4, replace=False)):
            ff, xx, yy = p[i] * norm_constant, x_test[i] * norm_constant,
                ↪ y_test[i] * norm_constant
            plt.subplot(subplots[plot_id])
            plt.grid()
            plot_scatter(range(0, backcast_length), xx, color='b')
            plot_scatter(range(backcast_length, backcast_length +
                ↪ forecast_length), yy, color='g')
            plot_scatter(range(backcast_length, backcast_length +
                ↪ forecast_length), ff, color='r')
        plt.show()

    return singular_loss
```

Here, an important point has to be made. As the *Numpy* array is being used along the code, it has to be mentioned, that *Numpy* library's operations are performed on CPU, which is a vital point, as calculation of weights performed on GPU have to be detached and then its' results have to be redefined (translated) into a *Numpy* array using the CPU. Thus indicating a certain slowdown of the whole process. Score function used in this entity is a MSE_Loss function provided by *PyTorch*. Additionally, as visible at the bottom of Listing 2.7, evaluation function has a possibility to draw 4 plots of randomly selected periods, showing how well generated curve is fitting the actual state of things. By default, this functionality is turned off to optimize training process.

2.10 Classifier

2.10.1 How to Run

To perform classification execution of the following command execution is required, along with directory containing input data files and another one prepared to receive output files.

Listing 2.8: Script execution using bash console

```
$ python3 driver.py input_path output_path
```

Where **driver.py** is the name of main script, **input_path** is the path to the directory containing input files and **output_path** is the path to the destination directory to which result files will be saved.

2.10.2 Classifier Input and Output

As the algorithm works as a set of scripts it is able to classify multiple signals in the same run. Therefore, a path to the directory containing pairs of ECG read binary files (with **.hea** and **.mat** extensions) provided in WFDB format as explained in section 2.2 is required as an input argument to the program. When in process, a single files are being open and read by the program and network adjusted chunks are being taken from them, as explained in section 2.8.

Outputs consist of the multiple **.csv** files which are saved in the directory indicated in the *output_path* parameter. Each file contains output for a single recording in a specific format presented below:

```
#<READ-ID>
AF I-AVB LBBB Normal PAC PVC  RBBB  STD STE
 0   1     0    0     0    0    0     0    0
0.88 1.0 0.283 0.67 0.60 0.461 0.237 0.751 0.0
```

First row of the output contains the ID of the file to results correspond. Second, contains class labels. Third contains an actual result of classification, meaning 0 or 1 corresponding to negative and positive result respectively. Forth contains partial results of classification, ranging from 0 to 1 as they went under normalisation min-max.
This output format was specified and required for the initial stages of PhysionetChallenge.

2.10.3 Classifier Description

Classifier used in this project was defined by it's author for the needs of **PhysionetChallenge / CinC2020**. The idea standing behind it's architecture was already mentioned above yet more thorough description might appear to be substantial, as it is the subject of this paper. Therefore, an implementation of **Model** class is presented below. It contains vital functions: *load* and *predict* being responsible for loading N-Beats trained networks and performing classification process consecutively.

Listing 2.9: Classifier implementation

```
class Model:
def __init__(self, device=torch.device('cpu')):
    self.forecast_length = 500
    self.backcast_length = 3 * self.forecast_length
    self.batch_size = 256
    self.classes = ['LBBB', 'STD', 'Normal', 'RBBB', 'AF', 'I-AVB', 'STE'
        ↪ , 'PAC', 'PVC']
    self.device = device
    self.nets = {}
    self.scores = {}
    self.scores_norm = {}
    self.scores_final = {}

def load(self):
    for d in self.classes:
        checkpoint = d + "_nbeats_checkpoint.th"
        net = NBeatsNet(stack_types=[NBeatsNet.GENERIC_BLOCK, NBeatsNet.
            ↪ GENERIC_BLOCK],
                        forecast_length=self.forecast_length,
                        thetas_dims=[7, 8],
                        nb_blocks_per_stack=3,
                        backcast_length=self.backcast_length,
                        hidden_layer_units=128,
                        share_weights_in_stack=False,
                        device=self.device)
        optimiser = optim.Adam(net.parameters())

        naf.load(checkpoint, net, optimiser)
        self.nets[d] = net

def predict(self, data, data_header):
    x, y = naf.organise_data(data, data_header, self.forecast_length,
        ↪ self.backcast_length, self.batch_size)
    for c in self.classes:
```

```
            self.scores[c] = naf.get_avg_score(self.nets[c], x , y)

        scores = list(self.scores.values())

        print(self.scores)
        max_score = max(scores)
        min_score = min(scores)
        result = {}
        for c in self.classes:
            self.scores_norm[c] = (self.scores[c] - min_score) / (max_score -
                ↪  min_score)
            self.scores_final[c] = 1 - self.scores_norm[c]
            result[c] = 0
            if self.scores_final[c] > 0.99:
                result[c] = 1

        return result
```

As visible in Listing 2.9 some indispensable parameters are set with default fixed values in the initializing function. Most of them is of N-Beats characteristic origin, and some are helping parameters facilitating the overall process.

Predict functions perform the final prediction itself. Based on the normalised scores returned by nine trained N-Beats networks it assigns labels to the highest scores, having a threshold of 0.99, where 1 value is a maximal possible score, which means that the distance between regressed curve made by a N-beats predictors from the original signal equals to 0. It return a set combining class label and the score obtained from the predictors. *Load* function does nothing else than loading 9 trained nets to an array and assign the array to the Model object itself.

3 Results

3.1 Training Results

The neural net descriptors were reaching different results in regards of evaluation loss due to the nature of signal associated wit ha certain class (cardiac disorder). Yet, in general each net definitely get better along the training process which will be best visible on graphs presented in the upcoming subsections. All graphs present data after normalization in the domain of time, where one unit of time equals 1/500 s.

3.1.1 Graphs Legend

Due to space limitations and scale in which the graphs are presented, I have decided to remove the legend on them and display it once in this section. Colors, marking, and meaning is consistent along all of the graphs presented below.

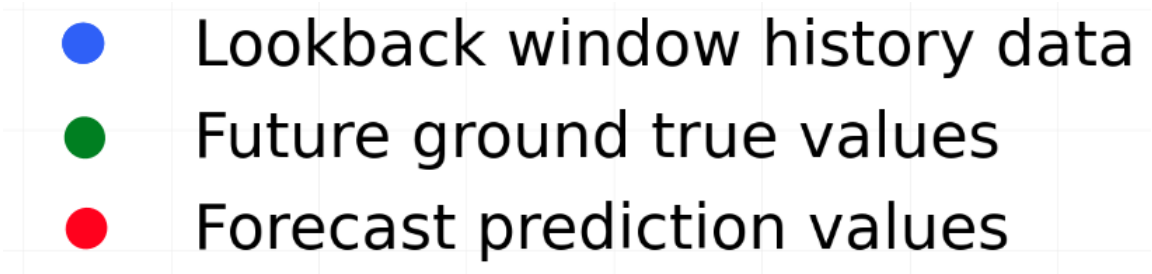

Figure 3.1: Graphs legend.

3.1.2 Network describing LBBB ECG signal

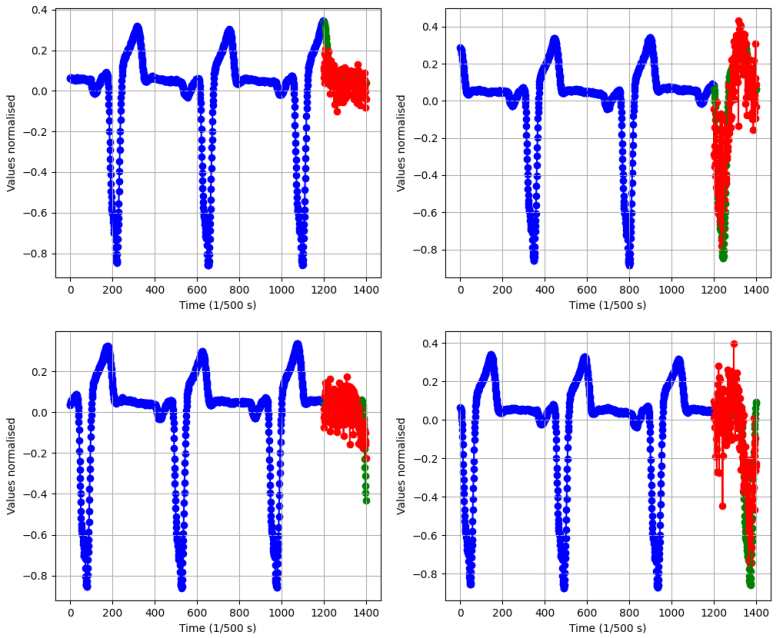

Figure 3.2: N-Beats describing LBBB signal after first epoch of training on examples from evaluation set.

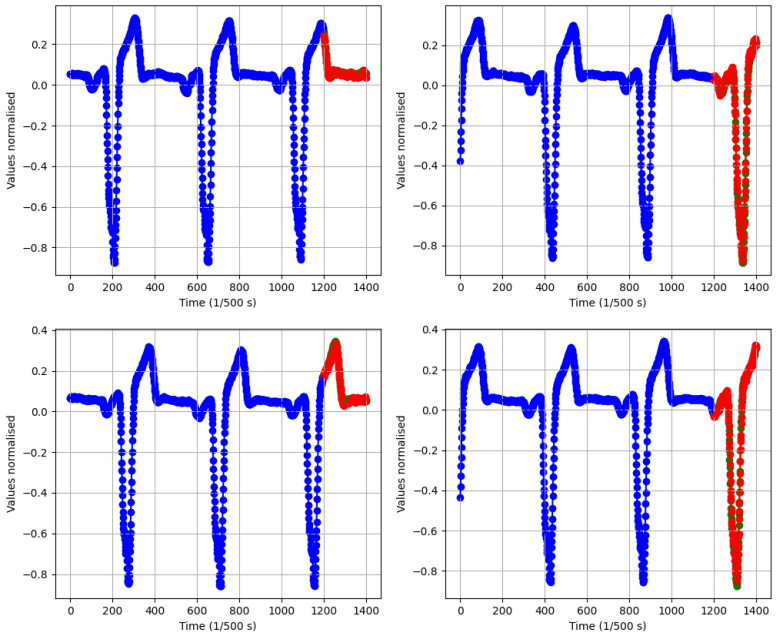

Figure 3.3: N-Beats describing LBBB signal after last (23rd) epoch of training on examples from evaluation set.

At the beginning of the training process, network is trying to predict forecast values based only on randomly set parameter values, and backcast history of length 1200 reads. As visible, it covers wide range of values somewhere around the proper values. Random values picks are visible, especially at the bottom right example, and predicted values look very noisy.

After proceeding with additional 22 epochs of training, neural net predictions are more clear and consistent. Trend and seasonality of LBBB specific ECG curve is visible, no noise can be detected on graphs and predictions fit tightly into expected range of values.

3.1.3 Network describing PVC ECG signal

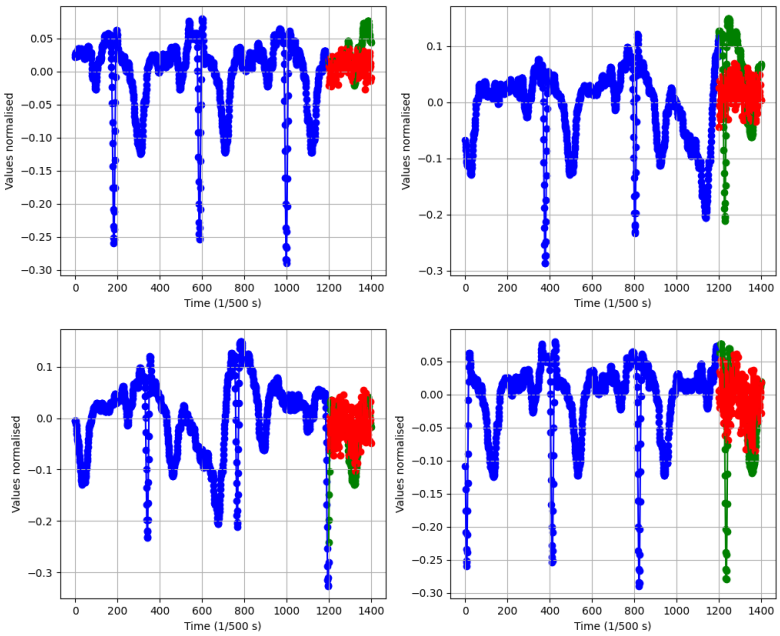

Figure 3.4: N-Beats describing PVC signal after first epoch of training on examples from evaluation set.

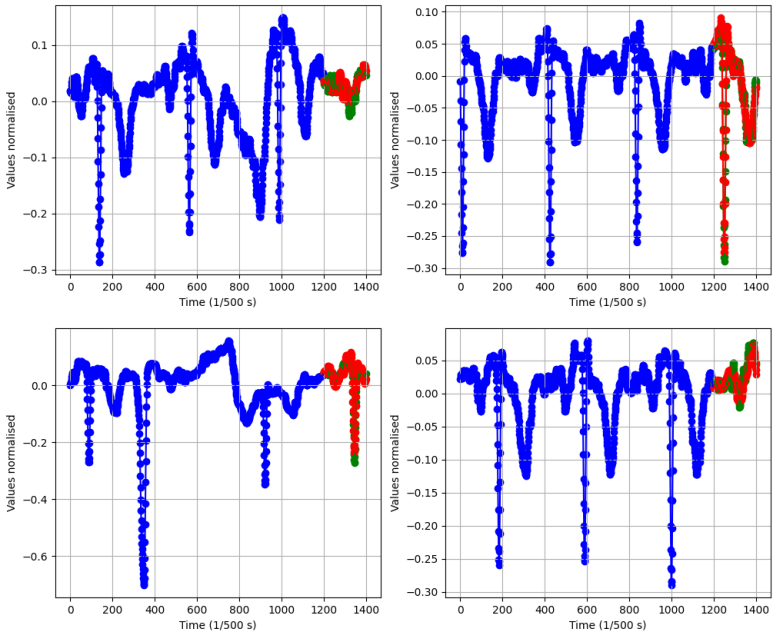

Figure 3.5: N-Beats describing LBBB signal after last (41st) epoch of training on examples from evaluation set.

PVC ECG signal was one of the most chaotic in the data set, which caused N-Beats having a hard time detecting its features and consistencies. At the beginning random values are being predicted somewhere around the average value of the lookback window. It is presented almost as a noisy red box of predicted values covering the green lines of the ground true.

Due to it's complexity, this signal required almost twice as much training time as the previous one, reaching it's peak of performance after 41st epoch. Still it has troubles with describing chaotic S-T segment and very quick QRS phase. However better trend depiction is visible and there is no noisy blocks covering average values range.

3.1.4 Network describing STD ECG signal

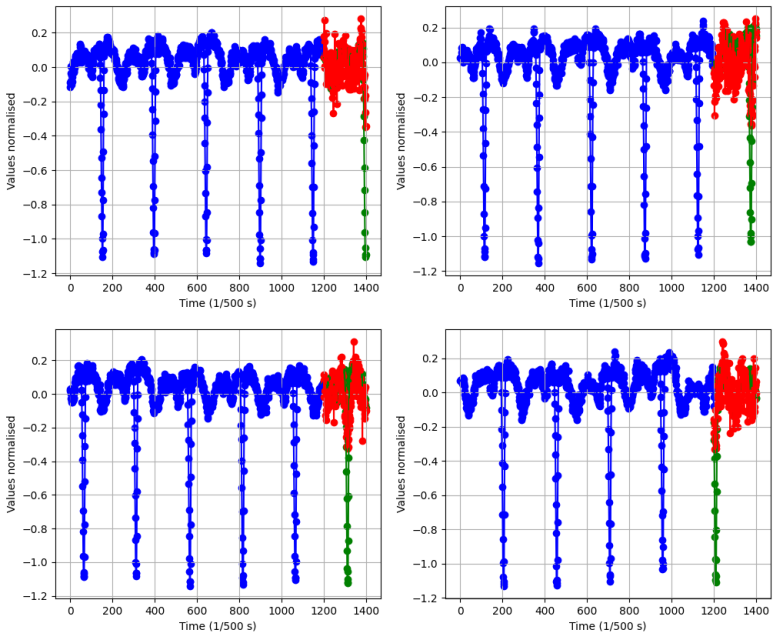

Figure 3.6: N-Beats describing STD signal after first epoch of training on examples from evaluation set.

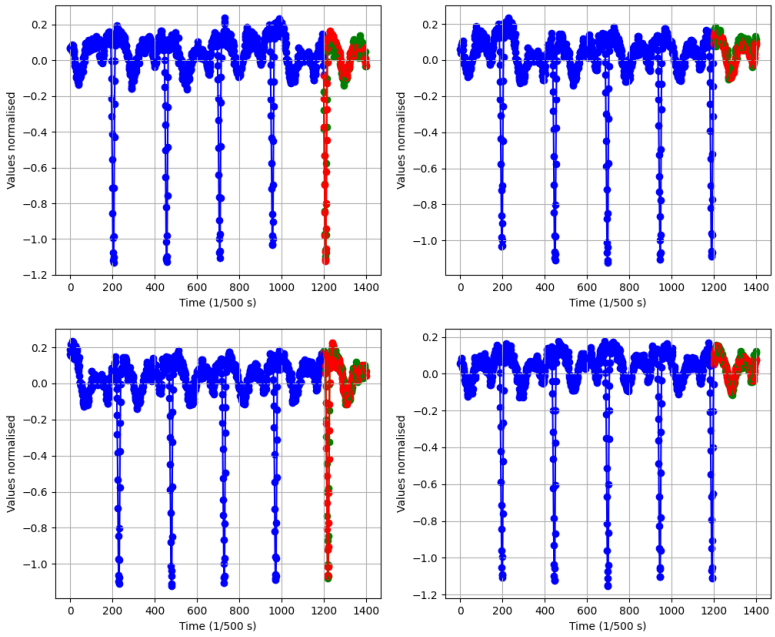

Figure 3.7: N-Beats describing STD signal after last (36th) epoch of training on examples from evaluation set.

STD signal reads visible in the exemplar section were recorded from a patient with higher heart beat rate, which affected the amount of picks fitting in the lookback window. Due to that, first predictions coming from neural network almost completely skipped QRS phase in their forecast. Even when the forecast window was starting from the Q-R hill, it was still omitted, and no values from the R-peak area were predicted.

After 36 training epochs neural net reached it's best performance, faithfully describing the trend of STD associated signal. However, values of forecast are slightly shifted which causes ground true line more visible. There is still some uncertainty left especially in the area of P-wave.

3.1.5 Network describing AF ECG signal

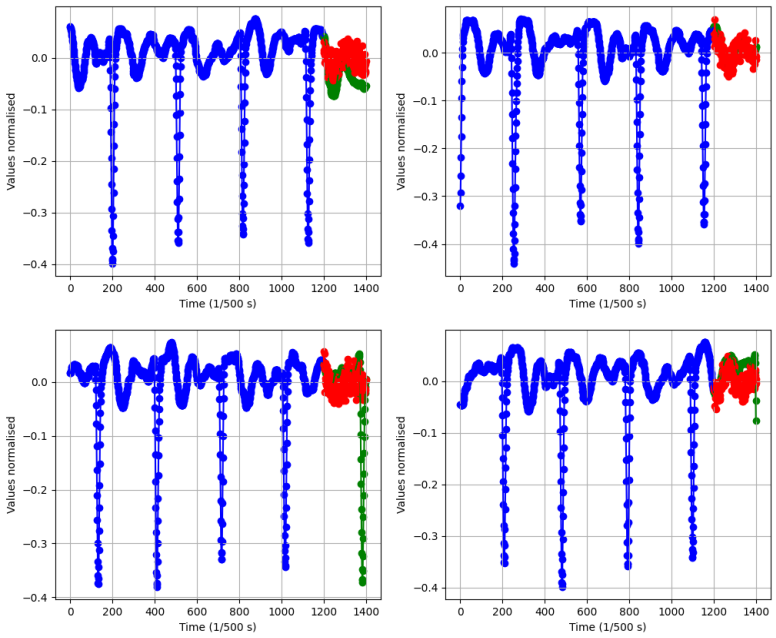

Figure 3.8: N-Beats describing AF signal after first epoch of training on examples from evaluation set.

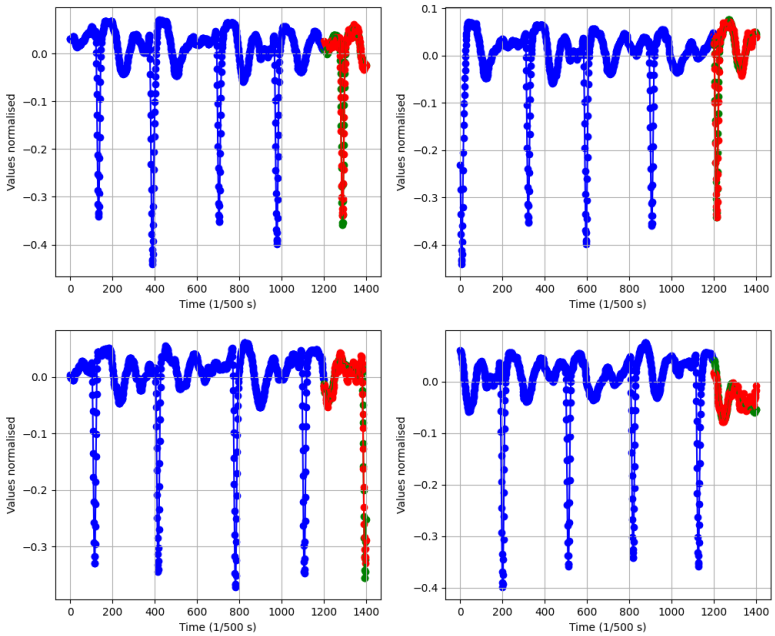

Figure 3.9: N-Beats describing AF signal after last (26th) epoch of training on examples from evaluation set.

For AF ECG signal, first prediction values were much more dense around average value of the T-wave / P-Wave area. Lack of ability to describe the trend is obvious and very visible, however in spite of the other methods presented it was not even trying to mimic the behaviour.

After 26 training epochs neural net reached it's best performance, however it was flattening the curve, often not reaching distant values in R-peaks or, as visible on the right bottom corner example, S-peak when it was at the beginning of the forecast window.

3.1.6 Network describing PAC ECG signal

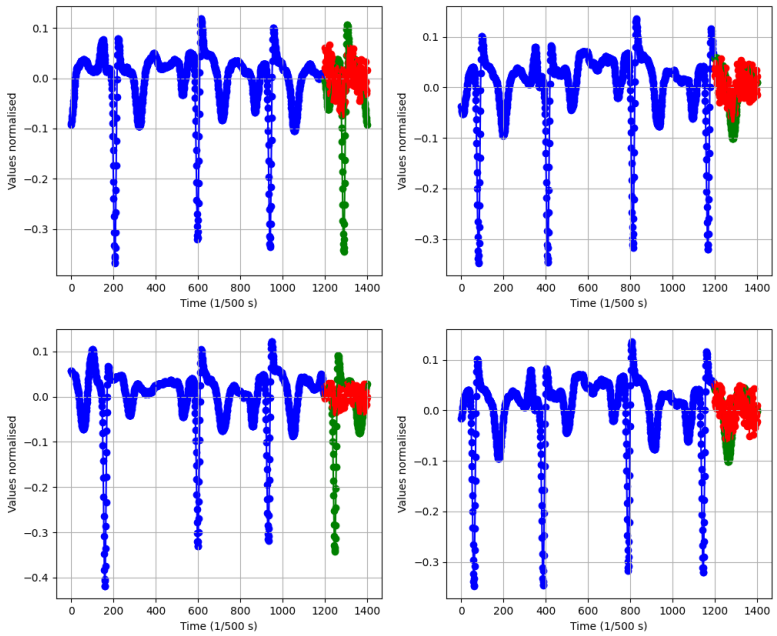

Figure 3.10: N-Beats describing PAC signal after first epoch of training on examples from evaluation set.

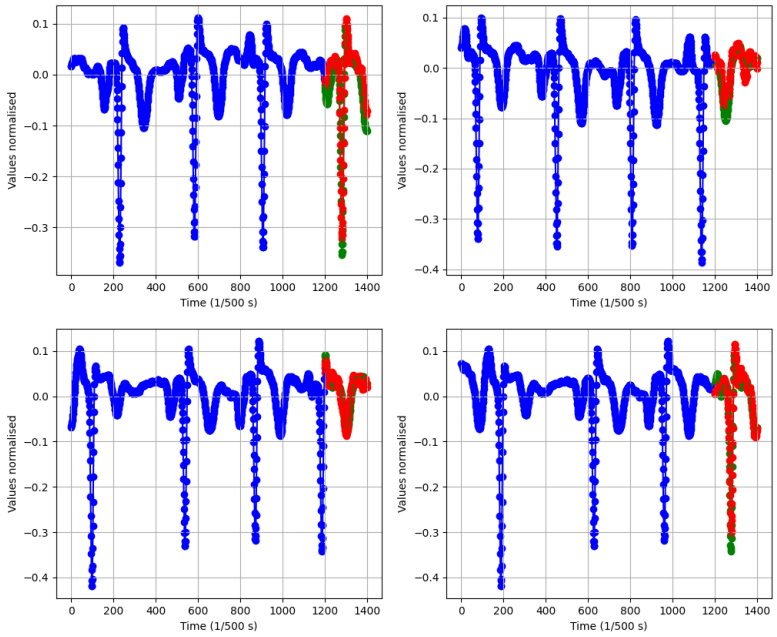

Figure 3.11: N-Beats describing PAC signal after last (19th) epoch of training on examples from evaluation set.

N-Beats also in case of PAC ECG signal ignored peaks and extreme values of QRS phase during his initial predictions. Yet, small curvatures are visible, thus indicating, that even without extensive training neural net was able to identify first behavioural patterns.

PAC describing network training was relatively quick, as it took only 19 epochs. After that period of time, neural net was available to depict and predict abnormal behaviours which is visible on the exemplar graphs.

3.1.7 Network describing Normal ECG signal

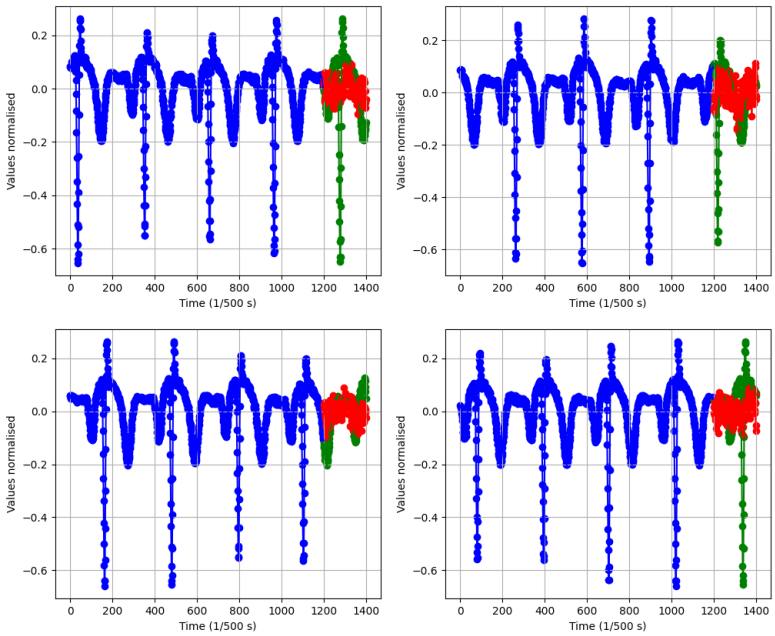

Figure 3.12: N-Beats describing Normal signal after first epoch of training on examples from evaluation set.

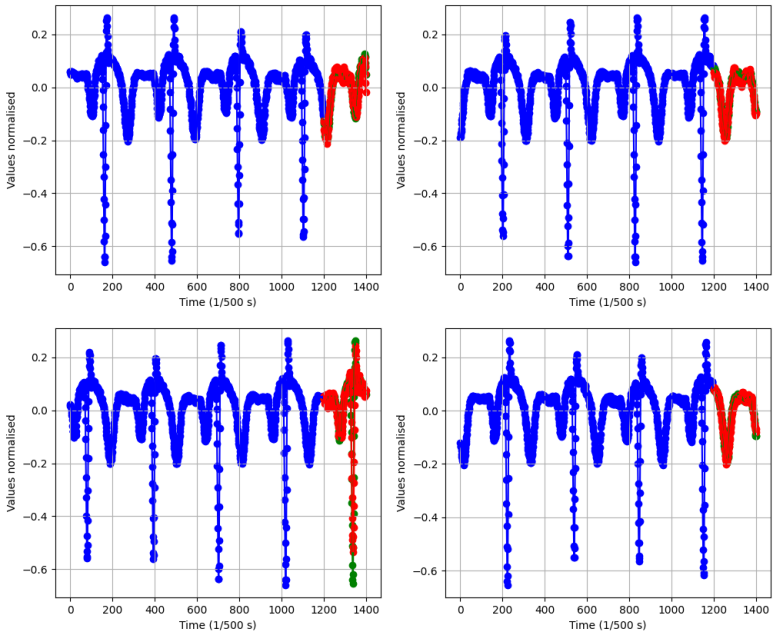

Figure 3.13: N-Beats describing Normal signal after last (19th) epoch of training on examples from evaluation set.

For Normal ECG signal initial N-Beats configuration was completely lost, not detecting any peaks from the QRS phase. It also skipped or left unnoticed waves from P or T segment, meaning it was just assigning close to average values for any new coming prediction.

Due to it's regularity this signal was described almost completely by the N-Beats neural network, appropriately forecasting all crucial points, however sometimes flattening values of the QRS peaks as visible on the example from the bottom left corner.

3.1.8 Network describing STE ECG signal

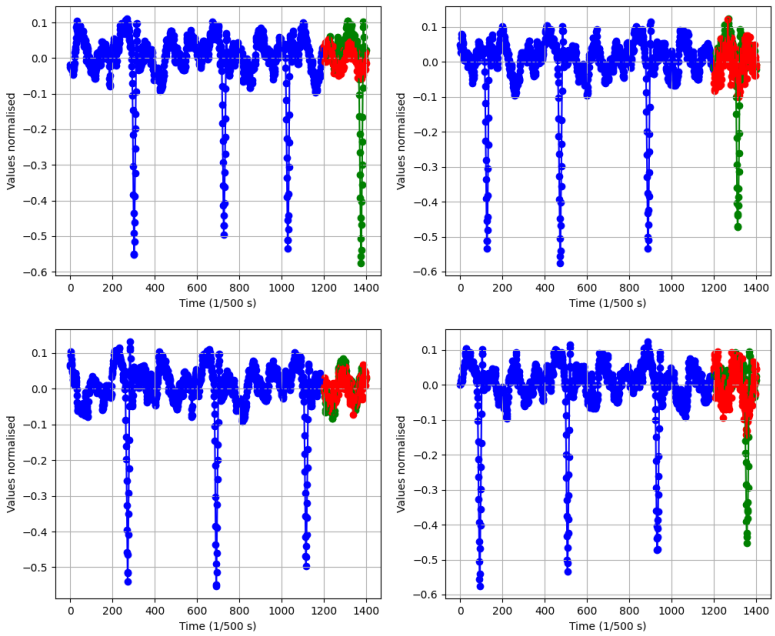

Figure 3.14: N-Beats describing STE signal after first epoch of training on examples from evaluation set.

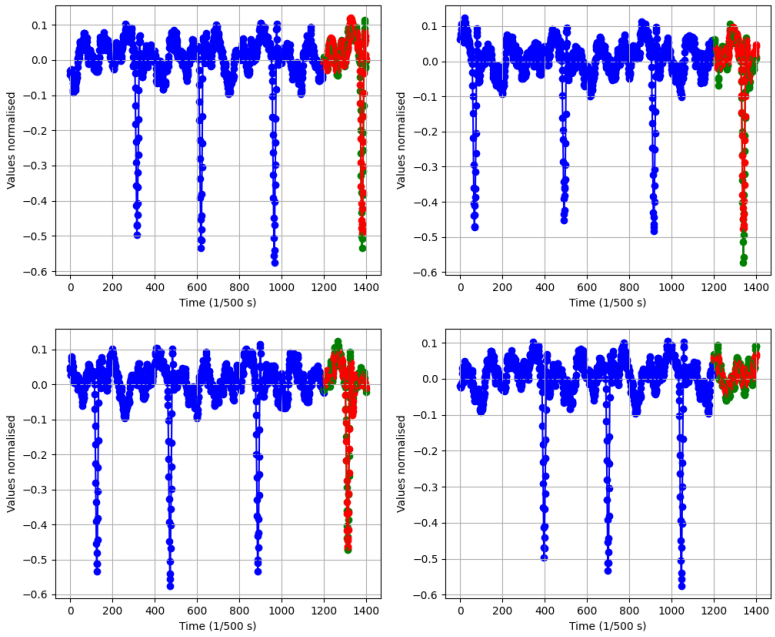

Figure 3.15: N-Beats describing STE signal after last (26th) epoch of training on examples from evaluation set.

As with most of the classes, initial configuration of the neural network returned results from the area of average lookback window values. No STE specific curvature is visible.

Due to noisiness of STE signal, N-Beats had a lot of troubles with describing it's characteristics. Some of the rapid jumps are not marked, which means that the neural net tried to somehow smoothen its predictions, making itself more generalized.

3.1.9 Network describing RBBB ECG signal

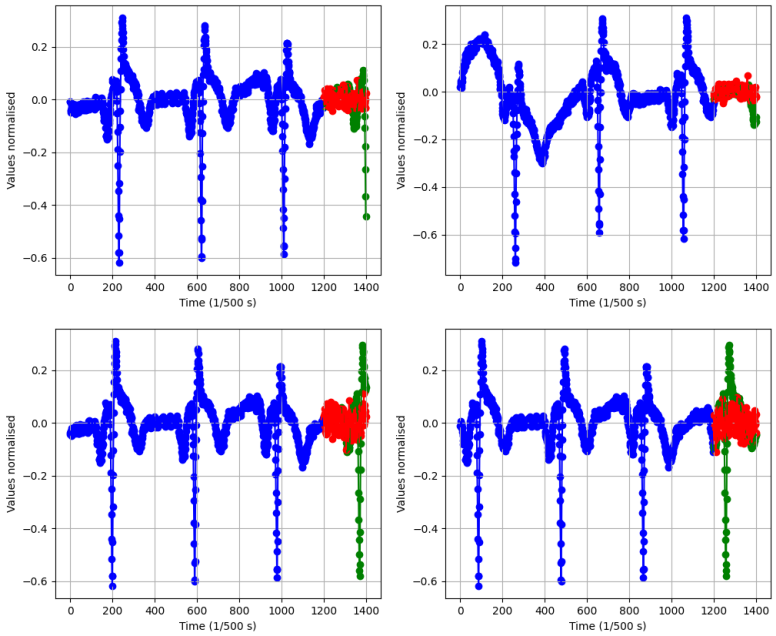

Figure 3.16: N-Beats describing RBBB signal after first epoch of training on examples from evaluation set.

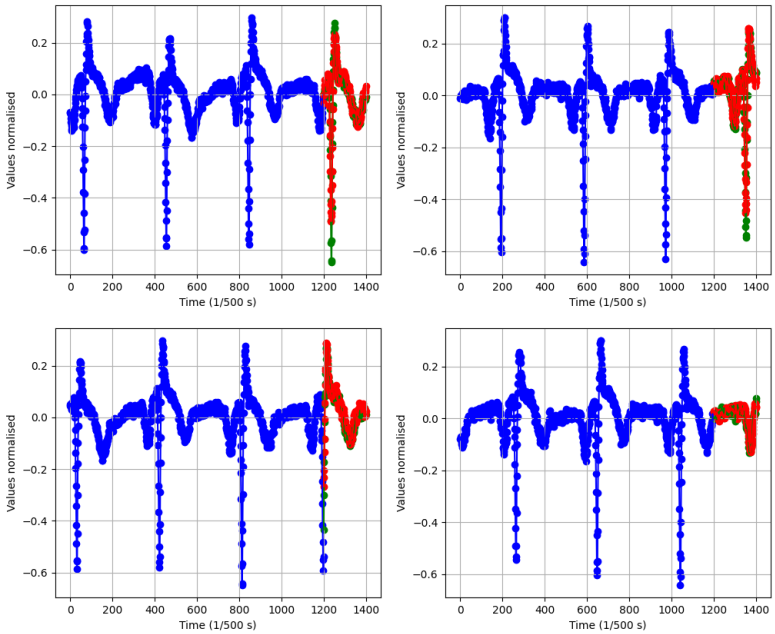

Figure 3.17: N-Beats describing RBBB signal after last (27th) epoch of training on examples from evaluation set.

For RBBB signal initial descriptor was not detecting any of the QRS points or important peaks, however, slight curvatures on the wave forms is visible at the top right example.

Although some edge values like R peaks are a little bit flattened, after reaching it's top performance after 27th training epoch N-Beats net was able to identify and properly mimic abnormal behaviour of RBBB signal as visible on the exemplar graphs.

3.1.10 Network describing I-AVB ECG signal

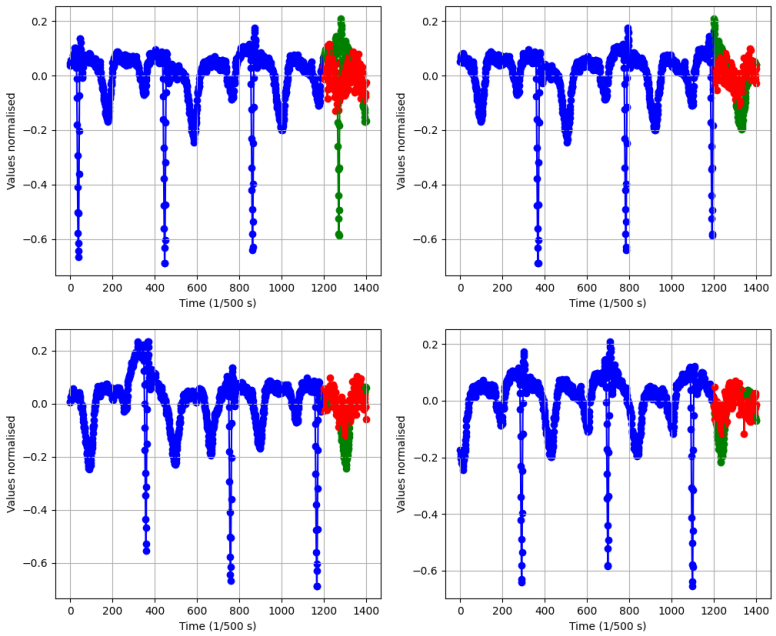

Figure 3.18: N-Beats describing I-AVB signal after first epoch of training on examples from evaluation set.

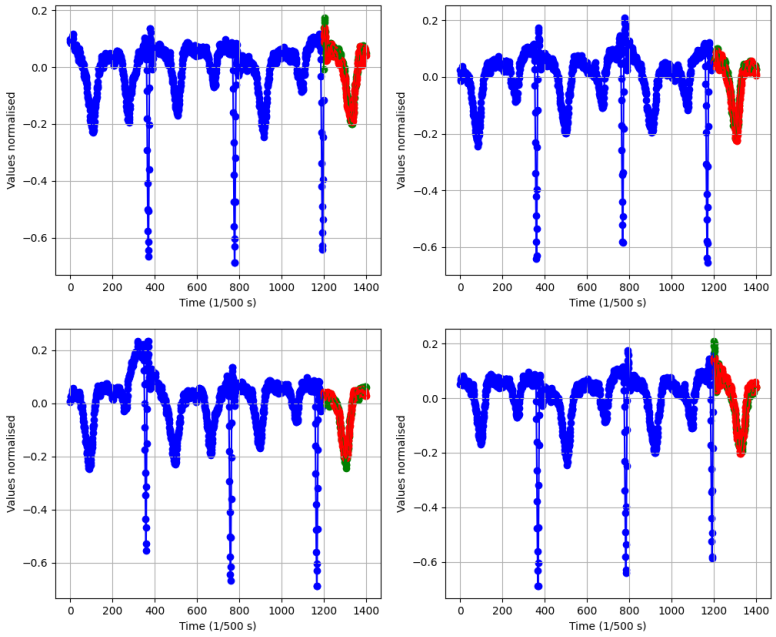

Figure 3.19: N-Beats describing I-AVB signal after last (21th) epoch of training on examples from evaluation set.

Although dominated by the average values forecast, initial neural network was able to identify main abnormality of the signal visible at exemplar graphs, keeping appropriate curvature on the drops in points Q and S.

After 21 epochs, descriptor of the I-AVB ECG signal was able to almost entirely match the forecast predictions with ground true signal, which means that it properly learned abnormalities of the given signal.

3.2 Classification Results

To obtain classification results and proper scores - equivalent to the ones used by the jury of the Challenge, a provided script was used. Script called *driver.py* consists of basic functions including *save_challenge_prediction* and *load_challenge_data* and the main function containing the loop which proceeds with the classification process. Aforementioned loop is a vital point of this script, therefore it is worth being presented:

Listing 3.1: Main loop of driver.py script

```
for i, f in enumerate(input_files):
    print('␣␣␣␣{}/{}...'.format(i+1, num_files))
    tmp_input_file = os.path.join(input_directory,f)
    data,header_data = load_challenge_data(tmp_input_file)
    current_label, current_score = run_12ECG_classifier(data,header_data,
        ↪ classes, model)
    # Save results.
    save_challenge_predictions(output_directory,f,current_score,
        ↪ current_label,classes)
```

As visible above, in each iteration a different input file is investigated and results of the investigation process are being saved into a separate output **.csv* file containing the final result of classification, labels and scores for each label.

3.3 Received Scores

3.3.1 Classifier trained on GPU with MSE loss function as decisive

Network parameters:

Table 3.1: Networks Parameters

Index	Parameter	Value
1	Forecast Length	500
2	Backcast Length	1500
3	Batch Size	256
4	Hidden Units	128
5	Blocks	3
6	Theta dimensions	[7,8]
7	Classifier decisive function	MSE loss

The very first results obtained were results of a training process, which took around 48 hours, and was performed on very few parts of training data-set (less 10% of one epoch). With that results visible in Table 3.2 were achieved.

Table 3.2: Scores after 48h training process

Index	Measure	Value
1	AUROC	0.671
2	AUPRC	0.088
3	Accuracy	0.779
4	F-measure	0.028
5	Fbeta-measure	0.033
6	Gbeta-measure	0.010

Yet, knowing that the training process was short and discovering leaks in the main loop everything was run again, trying to reach one-epoch-long training. This time the process was interrupted after approximately 96 hours of training - the reason for stopping the process was it being performed on a private machine, which was not a property of the main author and thus by him the stoppage was called (to avoid exploitation of the host's kindness). No measures were obtained.

More restriction were put on the training process, and with the same network architecture it was run for the third time. This try took approximately 72 hours to finish and covered around 20% of the training data-set - which is still a very low number. Yet, what could have been predicted, the results achieved were better - they are described in Table 3.3.

Table 3.3: Scores after 72h training process

Index	Measure	Value
1	AUROC	0.594
2	AUPRC	0.101
3	Accuracy	0.782
4	F-measure	0.044
5	Fbeta-measure	0.053
6	Gbeta-measure	0.017

Confusion matrix for this classifiers looks as follows.

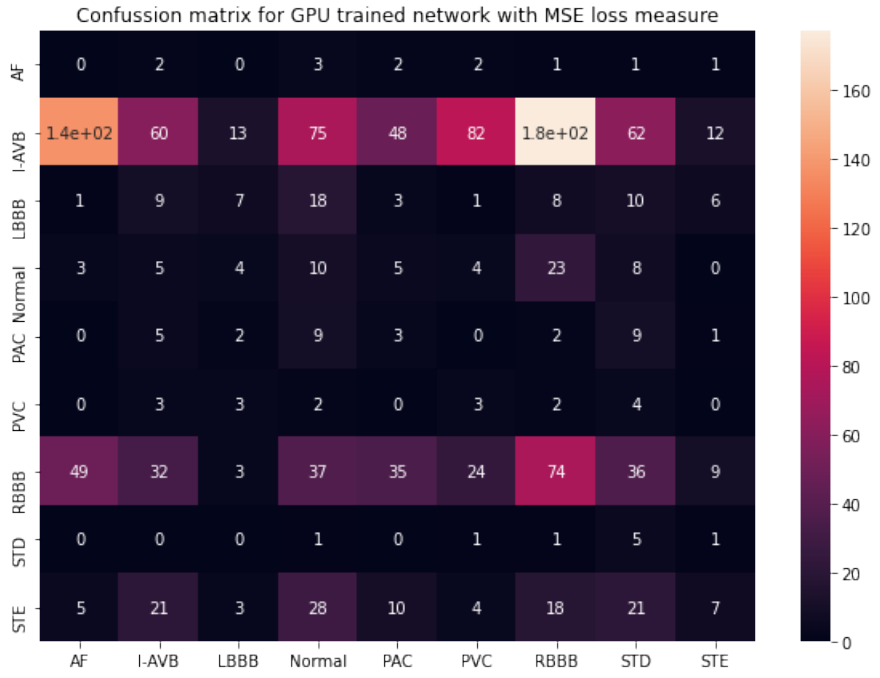

Figure 3.20: Confusion matrix for classifier based on big network trained on GPU with MSE loss measure.

On the confusion matrix we can see that this classifier almost always assigned only one label to any new signal presented. Higher number in the I-AVB row mean, that this class was assigned, especially when AF signal was provided or the RBBB one. For the AF originally labeled signals, out of 199 signals 140 was claimed to be of I-AVB label, and none of the signals was properly classified as AF. Out of 309 RBBB signals 180 were classified as I-AVB and only 74 was correctly labeled as RBBB It is also worth noting, that after performing well on training set, big neural network was absolutely unable to keep its ability to predict shapes and seasonality on testing set, resulting in following plots:

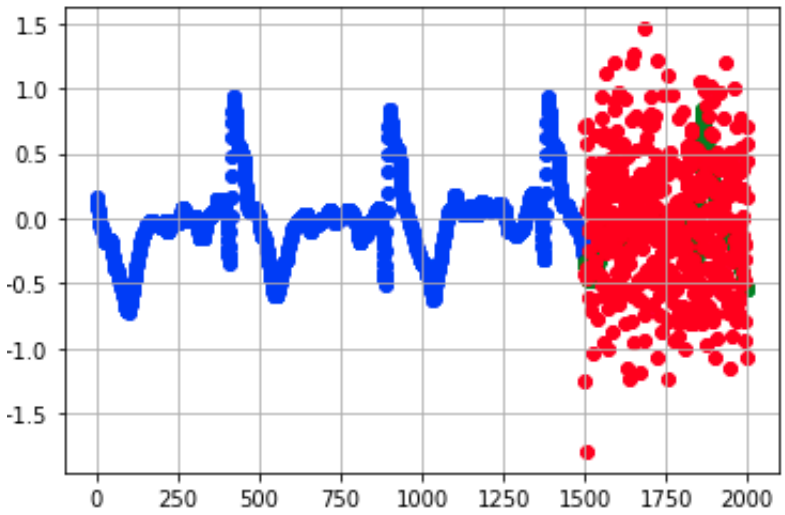

Figure 3.21: Huge neural network prediction on exemplar record from testing dataset

This happened because of the forecast length / backcast length ratio is to low - model has not had enough "memory" to properly adjust its prediction for so many points. Due to that it returned almost equally distributed points, paining box alike shape. To fix it, longer training process would be necessary along with extending lookback window size.

3.3.2 Classifier trained on CPU with L1 loss function as decisive

Network parameters:

Table 3.4: Networks Parameters

Index	Parameter	Value
1	Forecast Length	200
2	Backcast Length	1200
3	Batch Size	64
4	Hidden Units	16
5	Blocks	3
6	Theta dimensions	[7,8]
7	Classifier decisive function	L1 loss

After seeing that the results of the big neural network were miserable, decision was made to perform more experiments on simpler networks in order to be able to conduct required calculation on CPU. Surprisingly, the results of small networks were better than the one received after 48h of GPU huge neural network training, and just slightly worse than these received after 72h period. What is more, measures most important for the Physionet Challenge like

F, F-beta and G-beta measures improved. Measures scores are visible in the table below:

Table 3.5: Small network scores, CPU training

Index	Measure	Value
1	AUROC	0.622
2	AUPRC	0.0785
3	Accuracy	0.802
4	F-measure	0.068
5	Fbeta-measure	0.08
6	Gbeta-measure	0.028

On this confusion matrix we can see that this time LBBB label was almost always assigned when exposed to a new signal. Again, as the AF and RBBB signals are most numerous we can see that the vast majority of them was labeled as LBBB, however there is also a small shift in proportions visible, as now the distribution of the Normal and signals labels is narrower and the LBBB assignment is even more dominant. Out of 1276 predictions, to 762 a label LBBB was assigned. Out of 199 signals with original AF label, 170 was labeled by the classifier as LBBB.

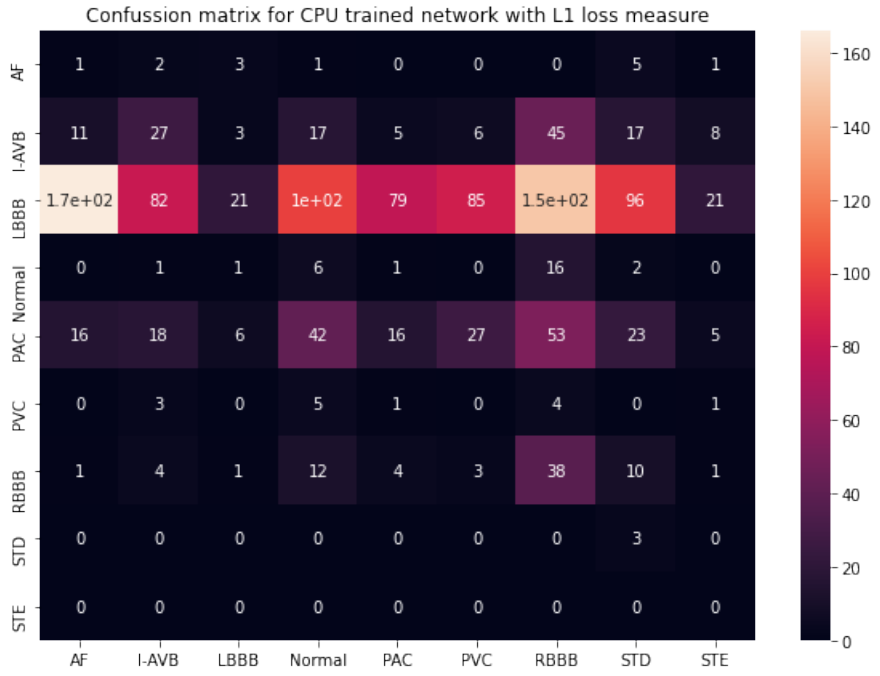

Figure 3.22: Confusion matrix for classifier based on small networks and L1 loss measure.

Surprisingly, smaller networks were also capable of remembering learned trends even when

exposed to testing dataset. Although in most cases patterns were chaotic, sometimes, when test signal was not noisy, proper network was able to generate almost perfect matches. It happened really seldom, however more often than in the bigger networks and it has a reflection in F-measure, F-beta measure and Gbeta measure scores.

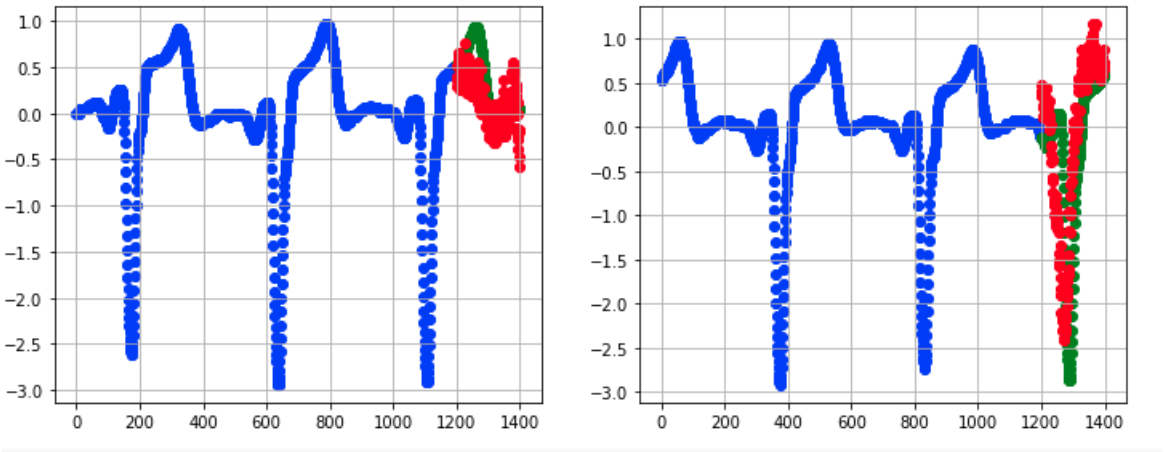

Figure 3.23: Small network trained to detect LBBB signal patter after being exposed to test set example

3.3.3 Classifier trained on CPU with MSE loss function as decisive

Network parameters:

Table 3.6: Networks Parameters

Index	Parameter	Value
1	Forecast Length	200
2	Backcast Length	1200
3	Batch Size	64
4	Hidden Units	16
5	Blocks	3
6	Theta dimensions	[7,8]
7	Classifier decisive function	MSE loss

Mean Squared Error function has an advantage of penalising more the distant predictions. This means, that in hypothetical situation, if one of the predictions has predicted a similar amount of points close to original curve and points which lay a certain distance from it, it will have rather more "negative" score, meaning that the algorithm put more emphasis on those distant ones. It is a contrary to L1 function, where the outcome score would be close to

being neutral. It resulted in a bit worse AUROC score at the level of almost 60% , however F measure, Fbeta measure and Gbeta measure were worse than in L1 case.

Table 3.7: Small network scores, CPU training, MSE loss function

Index	Measure	Value
1	AUROC	0.596
2	AUPRC	0.101
3	Accuracy	0.792
4	F-measure	0.07
5	Fbeta-measure	0.068
6	Gbeta-measure	0.0249

On a confusion matrix visible below it is certainly noticeable that network is almost always claiming to see LBBB pattern in a given signal. Out of 1276 predictions made, 903 were classified as a LBBB signal. It means that in 70.7 % of cases network returned LBBB class as a proper label. Again in the most numerous classes, only a few examples (if any) were labeled properly. No AF signals were classified as AF. only 6 out of 184 Normal signals were classified properly, and 31 out of 309 RBBB signal received their true value from proposed classifier.

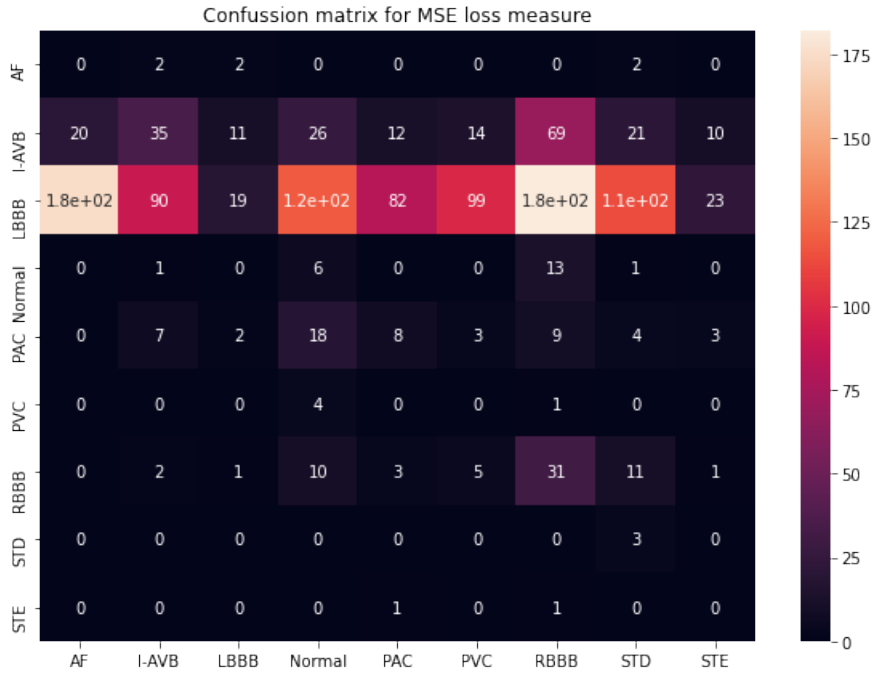

Figure 3.24: Confusion matrix for classifier based on small networks and MSE loss measure.

It was also visible on graphs, that LBBB descriptor were trying to create "noisy" predictions in order to cover the area of average values for most of the signal, which caused the median of MSE to be lower for this N-Beats network, than for ones which were trying to predict "clean" values and missed peaks and slopes.

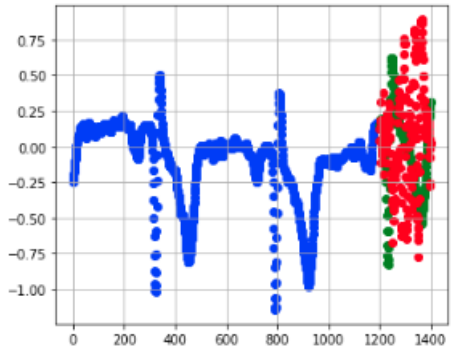

Figure 3.25: Small neural network describing LBBB signal trying to produce noisy forecast to cover average range of values

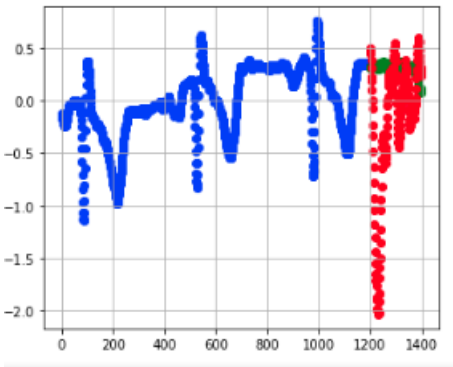

Figure 3.26: Small neural network describing STE signal trying to produce clean forecast with peaks in order to detect strong peaks

3.3.4 Classifier trained on CPU with MSE loss function as decisive with weight bias on LBBB descriptor

Network parameters:

Table 3.8: Networks Parameters

Index	Parameter	Value
1	Forecast Length	200
2	Backcast Length	1200
3	Batch Size	64
4	Hidden Units	16
5	Blocks	3
6	Theta dimensions	[7,8]
7	Classifier decisive function	weighted MSE loss

After seeing that the classifier is biased towards predicting LBBB labels, decision was made to penalise mistakes on LBBB signal forecaster 1.4 times more than on the others. The reason to do so was to check, if other descriptors would be able to put their label on signal if the decisive threshold was lowered due to lack of LBBB confidence.

Table 3.9: Small network scores, CPU training, weighted MSE loss function

Index	Measure	Value
1	AUROC	0.604
2	AUPRC	0.097
3	Accuracy	0.803
4	F-measure	0.097
5	Fbeta-measure	0.109
6	Gbeta-measure	0.0381

Although last three parameters improved due to bigger diversity in decision making, these improvements were minor, and surely not sufficient to call them satisfying. AUROC measure oscillates around 60% Visible trend from the previous experiment shifted to I-AVB class, making the classifier assign 650 label to I-AVB class, making it 50.9% of total assignments.

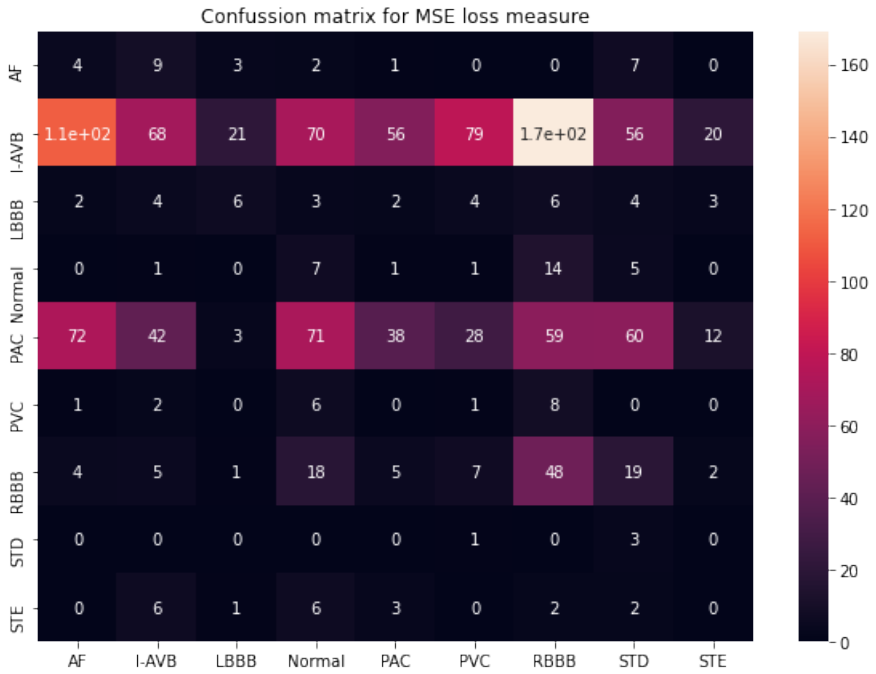

Figure 3.27: Confusion matrix for classifier based on small networks and weighted MSE loss measure.

It would be advisable also to test, whether if additional bias was put on I-AVB class would it have a direct impact on the measures, thus analogical experiment was performed with additional penalty of 1.2 on I-AVB class (meaning, that distance of bad predictions for I-AVB descriptor was multiplied 1.2 times). Following scores were achieved.

Table 3.10: Small network scores, CPU training, weighted MSE loss function with bias both on LBBB and I-AVB class descriptors

Index	Measure	Value
1	AUROC	0.618
2	AUPRC	0.091
3	Accuracy	0.804
4	F-measure	0.106
5	Fbeta-measure	0.116
6	Gbeta-measure	0.0415

These were the best results achieved, regarding Fbeta and Gbeta measures, which are taken into consideration as the most important ones in regard of the Challenge.

Confusion matrix distribution shifted, this time labels were divided into three dominant classes LBBB, PAC and RBBB, which means there was an actual improvement in decision

making, allowing for a better diversity in voting. With that said, the majority of the AF signals was labeled as PAC, 75 RBBB signals were classified properly, and 52 PAC signals were also classified as their true label indicates.

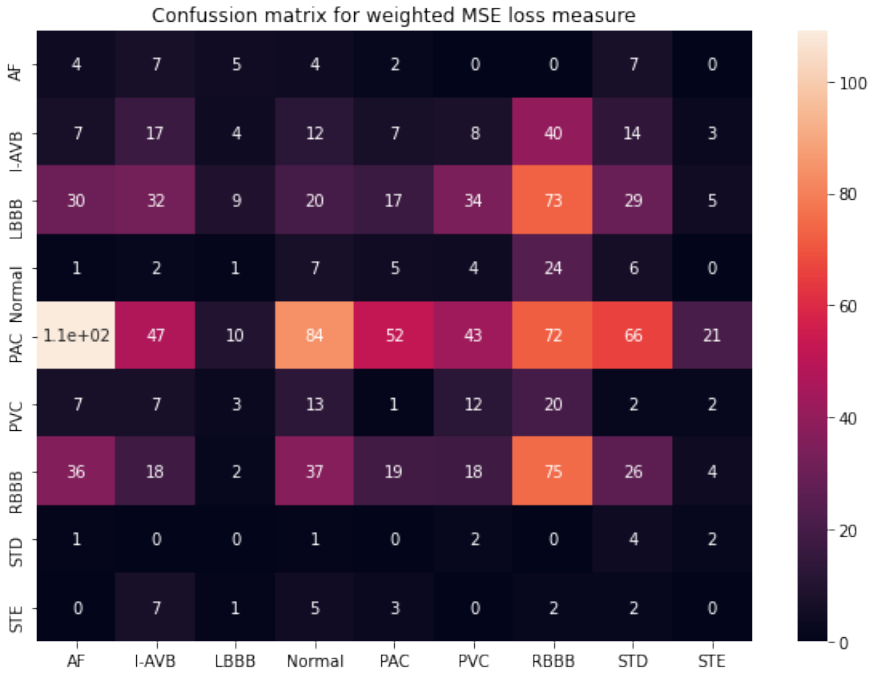

Figure 3.28: Confusion matrix for classifier based on small networks and weighted MSE loss measure with biased LBBB and I-AVB descriptors.

It is worth presenting the ROC curves of few selected classes. They represent the performance of the classifier for a certain class. They can be found below.

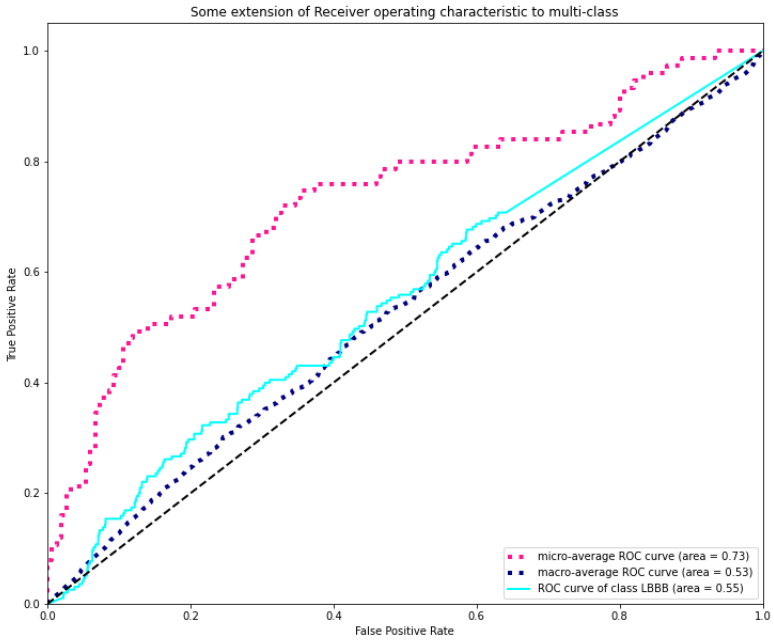

Figure 3.29: ROC curve for LBBB class.

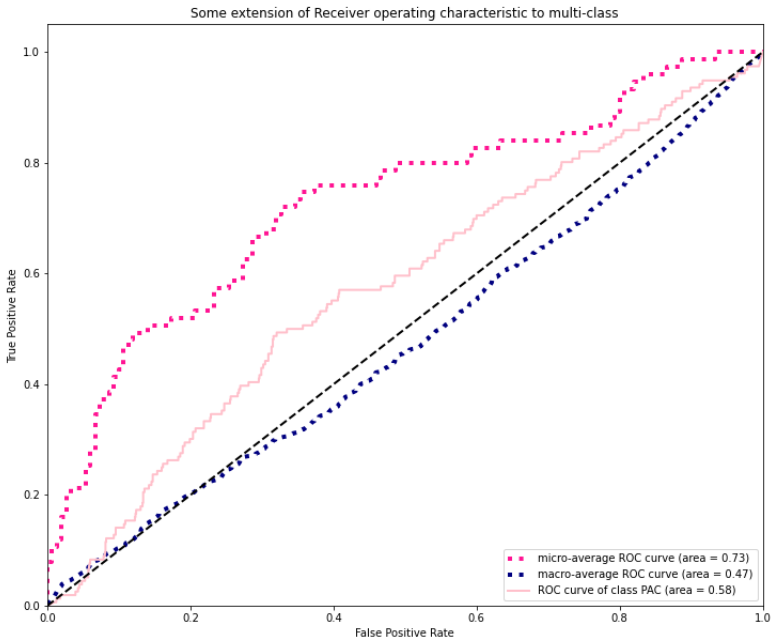

Figure 3.30: ROC curve for PAC class.

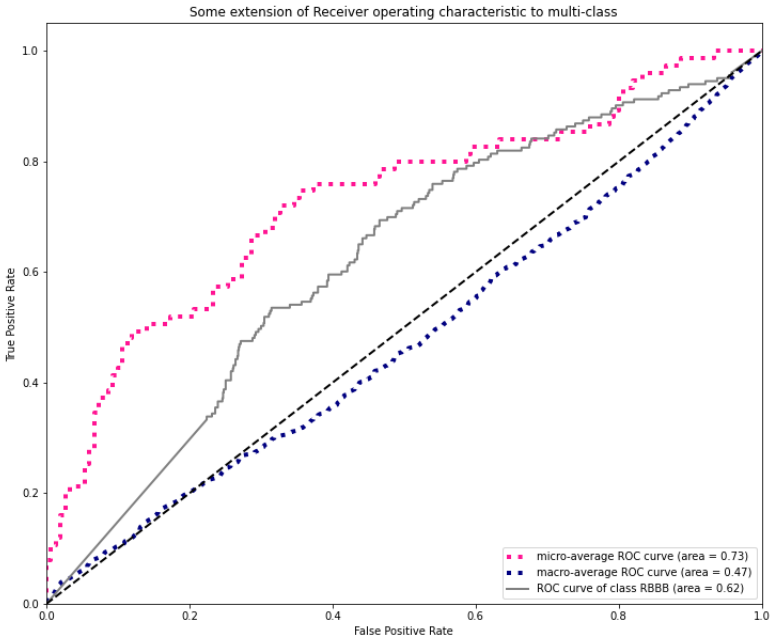

Figure 3.31: ROC curve for RBBB class.

4 Discussion

Results obtained with the use of N-Beats neural network may be claimed to be disappointing, especially the ones obtained from GPU training due to its time/effect ratio, as the measures provided by the Challenge (Fbeta measure and Gbeta measure) have reached numbers below 10% after the very first training. It might be slightly deceiving, as these measures penalise wrong, or not fully appropriate classification, much more heavily than rewarding the correct one. Area under the ROC curve is at the level of 0.329, which giving the fact of having 9 classification classes is not the worst one, but still is being far from good. Area under Precision-Recall curve is also small, which indicates, that our network has very poor precision. Yet, having an accuracy at the level of 77% the classifier might be claimed to have a potential, especially given that it was trained using only 10% of a singular epoch training data-set, which for neural network is almost unnoticeable.

The best experiment turned out to be the one performed on smaller network, with biased MSE loss function as decisive, with N-Beats neural networks trained on CPU with smaller batch size and little hidden units number. Among the experiments, the most noticeable phenomena was lack of support in assigning labels to some classes like: AF, STE, STD. Both during the first experiments without additionally penalised LBBB and in the later ones with extra penalisation these classes were almost completely omitted, which was visible in the presented confusion arrays. Also, a tendency to assign a single label in a vast majority of prediction was observed before implementing additional penalisation, which may indicate a strong influence of a lead characteristic towards the results of test signal exposure, by which the fact, that certain cardiac conditions are better exposed on certain leads is meant.

However, if one were to compare results obtained during the unofficial phase of PhysioNet / CinC Challenge 2020, and the best results reached during experiments performed during work over this project one may say, that objectively they are not the worst. Based on the results of the last experiment presented in the table 3.10 , receiving the best scores in measures considered in leaderboard of the PhysioNet challenge (Fbeta measure and Gbeta measure) (the PhysioNet/Computing in Cardiology Challenge 2020, 2020) a comparison was made in order to speculate possible place of this project in the unofficial ranking. Based on the unofficial leaderboard and taking best scores received into the consideration, final

place was calculated. It turned out that even though the classifier itself reached not the most astonishing results, it would be **placed at 332 position out of 396 works** (Challenge2020, 2020).

There are methods, which score even 98% of ECG classification accuracy. They are most often build over pattern recognition methods with strong preprocessing and independent component analysis (Yu & Chou, 2008). Some of them require patron figure to do a comparative classification. The features regularly used in ECG beat classification can be virtually divided into categories. The features can be acquired with time domain methods, with transformation methods, represented as statistical measures, etc. As to the pattern classification, efforts have also been devoted to the development of suitable classifiers for different kinds of feature sets, including linear discrimination or neural networks (Yu & Chou, 2008). This pattern proved to be much more efficient and useful than the one proposed by me in this work. Although the accuracy was the measure of the highest scores achieved, definitely comparative classification which used features or patron figure were more robust in the process. Regression proposed in this work has one strong flaw, it is very prone to the noises in signal and the differences in intervals. It was believed, that after learning on the training dataset neural networks will adjust to newly exposed signal, allowing proper regression performance only for the one network, which was dedicated to the original signal class. Yet, neural networks were successfully applied to classify ECG signal (Osowski & Tran Hoai Linh, 2001) - in the quoted example, fuzzy neural network classify given signal as ECG; and thus prove to be constantly worth investigation in the domain of more detailed classification. There were also different approaches to using neural networks as classifiers. Problem was divided into several bi-group classifier for each individual diagnostic class (Nugent et al., 1999). This allows for model ensemblance which is strongly advisable in regards of N-Beats neural networks (Oreshkin et al., 2019). Lack of this feature strongly affected abilities of the classifier proposed in this work, as only one layer of voting was performed based on MSE loss measure. Were there more bi-group classifier and more N-Beats nets to predict smaller chunks of information probably better results would be received.

Another factor having a direct impact on poor results of proposed morel is the input containing only one out of twelve available leads. More successful methods utilise the initial pre-processing module of beat detection which aims to locate each cardiac cycle in each of the recording leads and insert reference markers (Nugent et al., 1999). Other works also include multiple leads, at least focusing on lead-V1, V2, V3 and V4 (Chang et al., 2012). This

is mainly due to the fact, that each lead corresponded to different aspects of a heart activity and thus would facilitate detection of the trends which are not visible on a single lead due to its characteristics. This fact confirms the observations made during the experiments, as few of the classes are almost never classified: STE, STD and AF are seldom labeled, almost as if by a mistake. Including other leads will help to fix that behaviour.

However it has to be mentioned again, as this was a exploratory work and it's goal was to test whether the N-Beats architecture is suitable for classifying ECG signal and it's variations, that although the actual results are disappointing, a certain potential is already visible. Sadly, the grid search performed was as wide as the hardware allowed to in order to find most suitable hyper-parameters and define most fitting network blocks size. If the process were to be performed in the state of art way, an expensive computationally process would take surely more than 200 hours of training and evaluation and thus would require more advanced hardware and logging metrics to collect all the data. Yet, with adjustments to default settings N-Beats network proved that it is capable of performing it's regression on the ECG signal, having it's generators describe trend and seasonality, and of returning generic results which are useful in classification process itself.

5 Conclusion

Classification of cardiac disorders is a topic of the utmost importance, as by exploring it opportunities to save peoples live are being created. Not only do we, the community of data scientists, computer scientists and many many others, try to investigate the possibilities coming along with the technical development but also their usefulness in potentially high-risk scenarios. Having the most recent achievements in fields of medicine and computer science combined brings us, as a humanity another step closer to Utopian lifestyle, closer to becoming a species which freed itself from the dangers of the natural imperfections.
Goals set at the beginning of this paper were met, as all signals categories were visualised and described making them understandable and easier to interpret. N-Beats neural networks were implemented and the training processes, both more extensive using GPU and a quicker one using CPU, were performed. After that, new classifier model were proposed and certain experiments were perform to adjust and explore its behaviour. As the results were the outcomes of the experiments they were discussed along the process with the main discussion in the separate section. It might be surely claimed, that all the initial goals were met.

N-Beats neural network is one of the most recent achievements in the field of Artificial Intelligence and Machine Learning. Therefore the thesis formed under a question if N-Beats neural network is useful to regress an ECG signal and the results of this regression are useful for classification purposes was proved to be worth asking, and the investigation performed confirmed initial suspicions. Although mentioned network reached disappointing results in means of the PhysionetChallenge / CinC 2020 participation, only an initial investigation has been performed, and more tedious approach would be recommendable.
What was lacking in the whole experiment was the factor of independently run, long experiments which would allow for a better tuning of network parameters like the number of units in a single hidden layer, batch size, length of the look-back window, thetas dimensions or even the forecast length. All of the aforementioned parameters would directly influence the complexity of the network, thus resulting in either creating a neural network which would generalise regressed signal better, and remember certain trend and seasonality, or creating network which due to its size (number of units in hidden layer mainly influence this characteristic) would over-fit and although it would probably minimize error at the training data,

it would be lacking the ability to generalize properly, having evaluation error rate higher. Also it would be advisable to have a better model ensemblance for example by creating a multiple bi-grouping classifiers and to include multiple leads to have a more thorough insight into the domain signals. However it has to be emphasized that a new prediction model is being proposed in this work compared to classic classification methods (specially those based on feature extraction and pattern recognition). It is normal that lower results are obtained. But, this is a promising new concept that will require better refinement as it can provide solutions to vital problems, such as predicting whether a certain arrhythmia will end on its own (spontaneously) or, conversely, will require therapies such as electrical shock, anti-tachycardia pacing, among others. This would help to avoid unnecessary shocks or even certain therapies that result in a higher quality of life for the patient.

References

Adams, H. P., del Zoppo, G., Alberts, M. J., Bhatt, D. L., Brass, L., Furlan, A., Grubb, R. L., Higashida, R. T., Jauch, E. C., Kidwell, C., Lyden, P. D., Morgenstern, L. B., Qureshi, A. I., Rosenwasser, R. H., Scott, P. A., & Wijdicks, E. F. (2007). Guidelines for the early management of adults with ischemic stroke. *Stroke*, *38*(5), https://www.ahajournals.org/doi/pdf/10.1161/STROKEAHA.107.181486, 1655–1711. https://doi.org/10.1161/STROKEAHA.107.181486

Alper, A., Gungor, B., Turkkan, C., & Tekkesin, A. (2013). Symptomatic bradycardia caused by premature atrial contractions originating from right atrial appendage. *Indian pacing and electrophysiology journal*, *13*(3). https://doi.org/10.1016/s0972-6292(16)30628-3

Anumonwo, J. M., & Kalifa, J. (2014). Risk factors and genetics of atrial fibrillation [Atrial Fibrillation]. *Cardiology Clinics*, *32*(4), 485–494. https://doi.org/https://doi.org/10.1016/j.ccl.2014.07.007

Barnett, A. G., Dobson, A. J., & For the WHO MONICA (monitoring trends and determinants in cardiovascular disease) Project. (2004). Estimating trends and seasonality in coronary heart disease. *Statistics in Medicine*, *23*(22), https://onlinelibrary.wiley.com/doi/pdf/10.3505–3523. https://doi.org/10.1002/sim.1927

Bickerton, M., & Pooler, A. (2019). Misplaced ecg electrodes and the need for continuing training. *British Journal of Cardiac Nursing*, *14*(3), https://doi.org/10.12968/bjca.2019.14.3.123, 123–132. https://doi.org/10.12968/bjca.2019.14.3.123

Bun, S.-S., Latcu, D. G., Marchlinski, F., & Saoudi, N. (2015). Atrial flutter: more than just one of a kind. *European Heart Journal*, *36*(35), https://academic.oup.com/eurheartj/article-pdf/36/35/2356/17355513/ehv118.pdf, 2356–2363. https://doi.org/10.1093/eurheartj/ehv118

Burnett, J. (1985). The origins of the electrocardiograph as a clinical instrument. *Medical History*, *29*(S5), 53–76. https://doi.org/10.1017/S0025727300070514

C., G., & E., G. (2005). *A probabilistic interpretation of precision, recall and f-score, with implication for evaluation*. Springer, Berlin, Heidelberg. https://doi.org/https://doi.org/10.1007/978-3-540-31865-1_25

Chaitman, B. R., Zhou, S. H., Tamesis, B., Rosen, A., Terry, A. B., Zumbehl, K. M., Stocke, K., Takase, B., Gussak, I., & Rautaharju, P. M. (1996). Methodology of serial ecg classification using an adaptation of the novacode for q wave myocardial infarction in the bypass angioplasty revascularization investigation (bari). *Journal of Electrocardiology*, *29*(4), 265–277. https://doi.org/https://doi.org/10.1016/S0022-0736(96)80091-4

Challenge2020, P. /. C. (2020). Unofficial leaderboard for physionet / cinc challenge 2020.

Chang, P.-C., Lin, J.-J., Hsieh, J.-C., & Weng, J. (2012). Myocardial infarction classification with multi-lead ecg using hidden markov models and gaussian mixture models. *Applied Soft Computing*, *12*(10), 3165–3175. https://doi.org/https://doi.org/10.1016/j.asoc.2012.06.004

Cleland, P. N. M. T. L. A. C. M. M. M. C. (1991). *Apparatus and method for antitachycardia pacing using a virtual electrode* (US5181511A).

Gerstenfeld, E. P., & Marco, T. D. (2019). Premature ventricular contractions. *Circulation*, *140*(8), https://www.ahajournals.org/doi/pdf/10.1161/CIRCULATIONAHA.119.040015, 624–626. https://doi.org/10.1161/CIRCULATIONAHA.119.040015

Kligfield, P., Gettes, L. S., Bailey, J. J., Childers, R., Deal, B. J., Hancock, E. W., van Herpen, G., Kors, J. A., Macfarlane, P., Mirvis, D. M., Pahlm, O., Rautaharju, P., & Wagner, G. S. (2007). Recommendations for the standardization and interpretation of the electrocardiogram. *Journal of the American College of Cardiology*, *49*(10), https://www.onlinejacc.org/content/49/10/1109.full.pdf, 1109–1127. https://doi.org/10.1016/j.jacc.2007.01.024

Kosub, S. (2019). A note on the triangle inequality for the jaccard distance. *Pattern Recognition Letters*, *120*, 36–38. https://doi.org/https://doi.org/10.1016/j.patrec.2018.12.007

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*. https://doi.org/10.1016/j.ijforecast.2018.06.001

N-beats: Neural basis expansion analysis for interpretable time series forecasting implementation in pytorch by @philipperemy (philippe remy). (n.d.).

Nugent, C., Webb, J., Black, N., Wright, G., & McIntyre, M. (1999). An intelligent framework for the classification of the 12-lead ecg. *Artificial Intelligence in Medicine*, *16*(3), 205–222. https://doi.org/https://doi.org/10.1016/S0933-3657(99)00006-8

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting.

Osowski, S., & Tran Hoai Linh. (2001). Ecg beat recognition using fuzzy hybrid neural network. *IEEE Transactions on Biomedical Engineering*, *48*(11), 1265–1271.

Pollehn, T., Brady, W. J., Perron, A. D., & Morris, F. (2002). The electrocardiographic differential diagnosis of st segment depression. *Emergency Medicine Journal*, *19*(2), https://emj.bmj.com/content/19/2/129.full.pdf, 129–135. https://doi.org/10.1136/emj.19.2.129

the PhysioNet/Computing in Cardiology Challenge 2020. (2020). Classification of 12-lead ecgs: The physionet/computing in cardiology challenge 2020. https://doi.org/https://physionetchallenges.github.io/2020/

Yu, S.-N., & Chou, K.-T. (2008). Integration of independent component analysis and neural networks for ecg beat classification. *Expert Systems with Applications*, *34*(4), 2841–2846. https://doi.org/https://doi.org/10.1016/j.eswa.2007.05.006