CentraleSupélec In partnership with Inserm

PhysioNet :Classification of heart diseases using 12-leads ECGs

Research Internship

supervisors :
CentraleSupélec:
M. Jeremy Fix
M. Michel Barret
Inserm :
M. Julien Oster
M. Pierre Aublin
Ensta:
M. Franchi Gianni

Author :
Mouin Ben Ammar

2A Advanced techniques

Abstract

Objective: Cardiovascular diseases (CVD) are a major threat to health and one of the primary causes of death globally. The 12-lead Electrocardiogram (ECG) is a cheap, widely used and commonly accessible medical tool to detect cardiac abnormalities. Accurate diagnosis will allow for a proper treatment and an early intervention to prevent severe health complications.
The Objective of the PhysioNet/Computing in Cardiology Challenge 2021 is to develop an algorithm that automatically identifies 26 ECG abnormalities from 12-lead, 6-lead , 4-lead, 3-lead or just two leads from the 12-lead ECG recordings.
Our approach to the problem was firstly to apply a series of pre-processing methods to reduce the randomness of the data.
Secondly, the data will be passed through a convolutional neural network to extract the features , these convolutions are considered as the embedding layer to our third step wich is to take the output of this convolution and pass it through a multi-head transformer to make the final prediction. We introduce a DiceBCELoss to handle the problem of class imbalance, and thus improve the model's generalizability.
Main Results:
An entry was submitted to the official phase of the Physionet Challenge 2021. Using the challenge's metric, scores of 0.587, 0.574, 0.574, 0.577 and 0.572 on 12-lead, 6-lead, 4-lead, 3-lead, 2-lead validation data sets were obtained respectively.
Results on the challenge's hidden test set are expected by end of September after the conference Computing in Cardiology 2021.

Contents

0.1 List of Acronyms

ECG : Electrocardiogram
CVD : Cardiovascular diseases
SE : Squeeze-excitation
SVM : Support-vector machine
RBF : Radial basis function
RNN : Recurrent neural network
CNN : Convolutional neural network

Chapter 1

Introduction

1.1 Presentation

According to the World Health Organisation CVDs are the first cause of death worldwide , taking around 17.9 million lives each year [1]. "The annual burden of CVD on the European economy is estimated to be 210 billion" [2] each year.

ECGs are used to monitor and diagnose the heart. A heart produces electrical impulses which spread through the heart muscle. These impulses are detected by an ECG machine that displays these data as time series. Medical practitioners then interpret them to give their diagnoses.

Due to high mortality rate of heart diseases, early and accurate diagnosis of ECG abnormalities can prevent serious complications such as sudden cardiac death and improve treatment outcomes. It will become even more useful and relevant in places, where there is a lack of medical expertise. This motivates the requirement for a reliable, automatic, and low-cost system for monitoring and diagnosis,.

Recently, there has been a great attention towards accurate categorisation of heartbeats using deep learning methods. For instance , abnormalities detection models based on ECG signal as the 21-layer 1D convolutional recurrent neural network to detect atrial fibrillation, which was trained on single-lead ECG data [3], or the ensemble deep learning model for automatic classification of cardiac arrhythmia's based on single lead ECGs, which fused the decisions of ten classifiers and outperformed the single deep classifier [4].

The previous works focus on at most 9 ECG abnormalities's.

Hence, a more generalizable and accurate algorithm for automated ECG pattern classification that is able to adapt the complexity and difficulty of ECG's interpretation is highly desirable [5].

1.2 Project description

This work is based on the PhysioNet/Computing in Cardiology Challenge 2021 [6].

Physionet is a research resource for complex physiologic signal, it is a repository of free medical research data and related open-source software. It is managed by the MIT laboratory for computational physiology. One of its well known data set is the MIT-BIH Arrhythmia Database [7]. Computing in Cardiology is an international scientific conference held annually.

In cooperation with the annual Computing in Cardiology conference, Physionet hosts an annual challenge, focusing research on unsolved problems in cardiology.
This year challenge aims at developing an open source algorithm that identify heart diseases from a set of $\{2, 3, 4, 6, 12\}$-leads of ECG recordings with a vast unbalanced and non uniform data sets. Previous cardiology-based challenges were using the whole 12 ECG leads, and before that only one lead, but for the first time the 2020 challenge data consists of different combination of ECGs leads to identify the importance of each type of lead and how many are needed in order to make a precise classification. Using the 12-leads ECGs repositories provided by Physionet Challenges, we aim at developing an algorithm to create an accurate and automatic diagnostic system for a variety of cardiac diseases. The goal of our project is also to reach conclusions on which algorithm is better and what type of inputs should be used.

Problematic and goals :

- Developing a robust model that is able to identify 26 types of ECG abnormalities by analysing 12-lead ECG signals from multiple non uniform data sets.
- Identifying which prepocessing and which classification methods are better.
- Discussing which type of input is better to accurately classify these heart diseases.

The public challenge data consists of over 100,000 12-lead ECG signals from 6 different countries, from which 88,000 was shared as public training data , 6,630 ECGs retained as public test data, and 16,630 ECGs retained privately as test data for the final result.

1.3 Contribution

First, since this work is a follow up on the the final year project made by the CentraleSupélec students "Nina Achache" , "Mélina Benahmed" and "Aymane El Hichami" my first job was to read and understand their work in order to use the successful experiences they made and neglect the unuseful from the start.

After that my main mission was to learn about the transformer architecture in order to create one that is suited for our problem setup.
Since This year the input format varies between different number of leads,we chose to create a model that takes always a single lead as input,mainly to optimise the training time and to make the model more generalised , so I had to modify the different parts of the previous work (preprocessing,reading the data...) to be adapted with the current type of input since for last year version of the challenge there were a 12 ECG-leads input always.

Once this ground work was established, my mission shifted towards optimising the model, such as experiencing with different losses and try to find the best one for this problem,in which we found that the combination of the dice and BCE loss was the best fit.

In between those steps, there was always debugging work to do and correcting some of the model aspects to make it more fit for the challenge setup.

After establishing those steps, the rest of the work was mainly to experience with the different models to select the best one,
different preproceesing steps as well as the models parameters and global architecture in order to obtain the best challenge metric.

1.4 Literature Overview

The 12-lead ECG is an accessible and widely tool to screen for heart disease diagnoses [8].
The main sequential steps considered in ECG classification are using features extraction, normalisation, filtering and then classification. Two approaches are to be considered in this matter either a "same features as used by cardiologists" approach or a deep learning approach.

The clinical approach uses feature extraction, which describes the signal by morphological, temporal and statistical features. It is based completely on cardiologists expert knowledge and the extracted features (usually called hand crafted features).
Practitioners observe the P, Q, R, S, and T wave amplitudes and intervals to give their diagnoses.
Thus typical extracted features are, for instance, capturing information about the extrema of waveforms, RR intervals, QRS complex duration etc...
Figure 1.1 displays some detail on the ECG curve and morphological assessment of the wave and intervals [9].

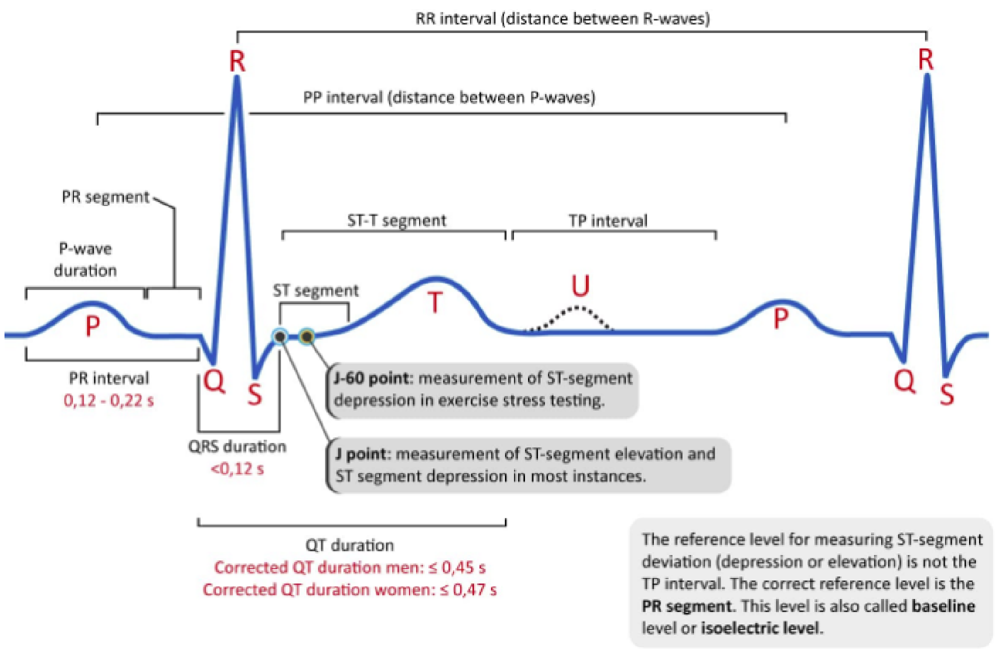

RR interval (distance between R-waves)
R
PP interval (distance between P-waves)
R
PR segment
P-wave duration
ST-T segment
TP interval
ST segment
T
U
P
P
PR interval
0,12 - 0,22 s
Q
S
J-60 point: measurement of ST-segment depression in exercise stress testing.
Q
S
QRS duration
<0,12 s
J point: measurement of ST-segment elevation and ST segment depression in most instances.
QT duration
Corrected QT duration men: ≤ 0,45 s
Corrected QT duration women: ≤ 0,47 s
The reference level for measuring ST-segment deviation (depression or elevation) is not the TP interval. The correct reference level is the PR segment. This level is also called baseline level or isoelectric level.

Figure 1.1 : Normal ECG characteristics [9]

However, a proper interpretation requires experienced clinicians to carefully examine and recognise pathological inter-beat and intra-beat patterns, besides, this process is time consuming and subject to discrepancy.
Therefore, an accurate algorithm for ECG abnormalities classification is highly desirable.

For the deep learning approach, the used techniques are as follows:

- The feature extraction techniques used by researchers are handcrafted features or transformation such as Continuous Wavelet Transform (CWT), Discrete Wavelet Transform (DWT), Discrete Cosine Transform (DCT), Discrete Fourier transform (DFT), Principal Component Analysis (PCA), Pan-Tompkins algorithm, Independent Component Analysis (ICA) etc.

- Techniques such as low pass linear phase filtering, linear phase high pass filtering are used for noise removal.
  For baseline adjustment, techniques such as median filter, linear phase high pass filter, mean median filter etc. are used.

- Classification techniques used are machine learning algorithms (on extracted features) SVM, Adaboost, Naïve Bayes, k-Nearest Neighbours and Random Forest or Deep Learning algorithms (transformer , Perceptron, RBF, CNN, RNN...) and sometimes a combination of machine learning and deep learning algorithms.

- For data normalisation techniques such as Z-score and Unity Standard Deviation (SD).

The winners of 2020 Physionet Challenge used a combination of the two methods [10].

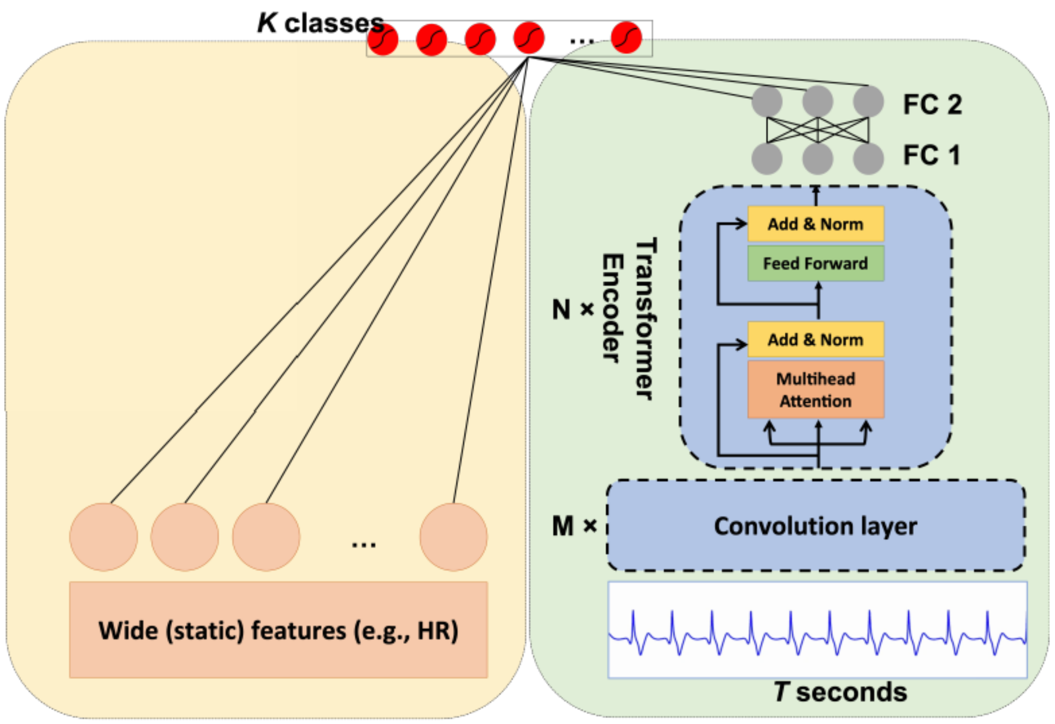


Figure 1.2 : Wide and Deep Network of the challenge's winners [10]

They extracted hundreds of handcrafted features using a Random Forest to scale the importance of each feature and then added a couple of static features.
The second component of the model consists of a transformer model [11] to classify the abnormalities using the multi-head attention mechanism.The complete model consists of 13,643,885 parameters.
They concluded that their architecture did not necessarily perform better than an RNN but the training time was considerably less and that the Noam [12] optimizer was crucial to maintain the transformer's performances.

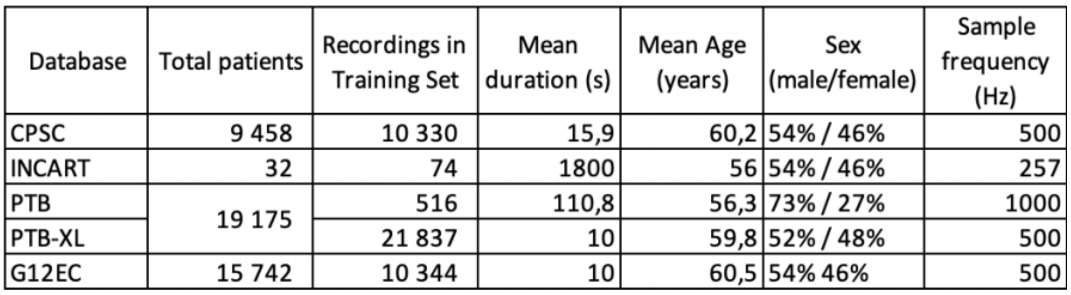
We chose to focus on deep learning methods as it is the best and most commonly used type of algorithms for such a complex task. The most popular architectures for the given task are recurrent neural networks (RNNs) and convolutional neural networks (CNNs), yet, since the introduction of the attention mechanism with transformers this type of architecture is proving to be superior in many types of deep learning problems [13],

Chapter 2

Data set

2.1 Data set description

The public challenge data consist of over 100,000 12-lead ECG signals from 6 different data sets originated from 4 countries across 3 different continents, including 2 data set2 from China, 2 data sets from Germany, 1 data set from Russia. 88,000 of these ECGs samples was shared publicly as training data , 6,630 ECGs retained privately as validation data, and 16,630 ECGs retained privately as test data. The main characteristics of the data sets(used in the 2020 version of the challenge and the 2021 version) provided for training are summarised in the table below [5].

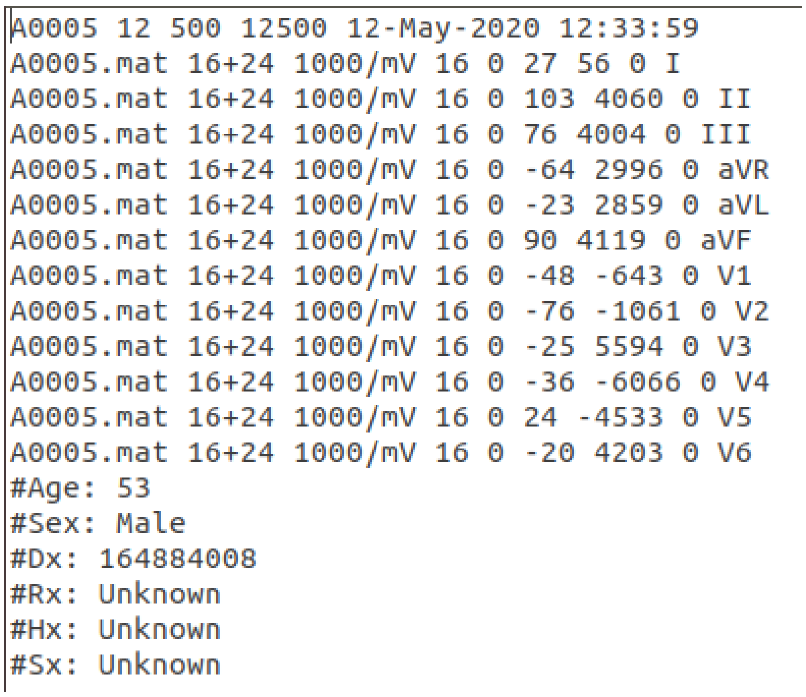


Database	Total patients	Recordings in Training Set	Mean duration (s)	Mean Age (years)	Sex (male/female)	Sample frequency (Hz)
CPSC	9 458	10 330	15,9	60,2	54% / 46%	500
INCART	32	74	1800	56	54% / 46%	257
PTB	19 175	516	110,8	56,3	73% / 27%	1000
PTB-XL		21 837	10	59,8	52% / 48%	500
G12EC	15 742	10 344	10	60,5	54% 46%	500

Figure 2.1 :Data Summary

the added databases in this year version are The Ningbo Data set with a total number of 45,152 recordings with a sampling frequency of 500Hz, and the CPSC2018 extra with a recordings and 500Hz as a sampling frequency.

To fully represent each recording 2 types of files are used. A .hea file that contains discrete information about the recording : sample frequency, number of samples, maximum amplitudes of each lead... plus some information about the patient: age, sex, diseases, surgeries...



```
A0005 12 500 12500 12-May-2020 12:33:59
A0005.mat 16+24 1000/mV 16 0 27 56 0 I
A0005.mat 16+24 1000/mV 16 0 103 4060 0 II
A0005.mat 16+24 1000/mV 16 0 76 4004 0 III
A0005.mat 16+24 1000/mV 16 0 -64 2996 0 aVR
A0005.mat 16+24 1000/mV 16 0 -23 2859 0 aVL
A0005.mat 16+24 1000/mV 16 0 90 4119 0 aVF
A0005.mat 16+24 1000/mV 16 0 -48 -643 0 V1
A0005.mat 16+24 1000/mV 16 0 -76 -1061 0 V2
A0005.mat 16+24 1000/mV 16 0 -25 5594 0 V3
A0005.mat 16+24 1000/mV 16 0 -36 -6066 0 V4
A0005.mat 16+24 1000/mV 16 0 24 -4533 0 V5
A0005.mat 16+24 1000/mV 16 0 -20 4203 0 V6
#Age: 53
#Sex: Male
#Dx: 164884008
#Rx: Unknown
#Hx: Unknown
#Sx: Unknown
```

Figure 2.2: .hea file

The second file is a .mat signal that consists of a $12 \times N$ matrix where N is the number of samples. Below the 12-leads ECG signal represented by this matrix.

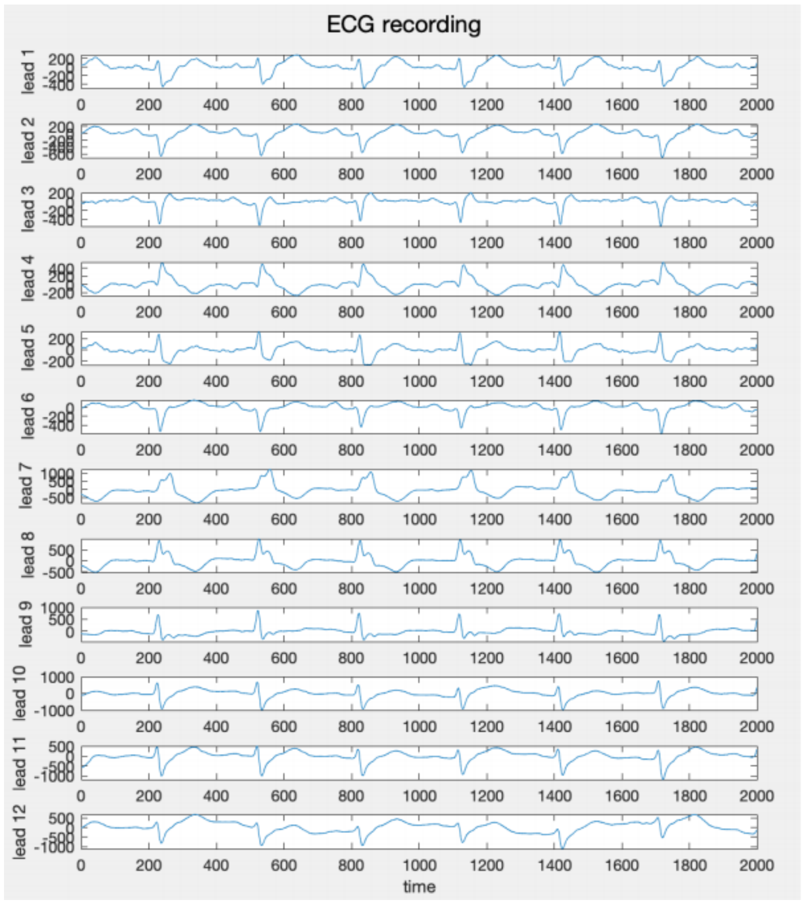


Figure 2.3: .mat file representing the 12-leads of an ECG

The sampling frequency of the signals varies from 257 Hz to 1,000 Hz, and the length of the signals varies from 6 seconds to 30 minutes. There are 133 labelled abnormalities(diseases + normal class). The same type of abnormality may be represented by multiple labels. Out of these 133 abnormalities, only 30 are to be scored and considered by The challenge, and since 8 of these abnormalities are to be treated as equivalent (two at a time), therefore, the final prediction consists of only 26 classes, those classes are quite unbalanced as shown below.

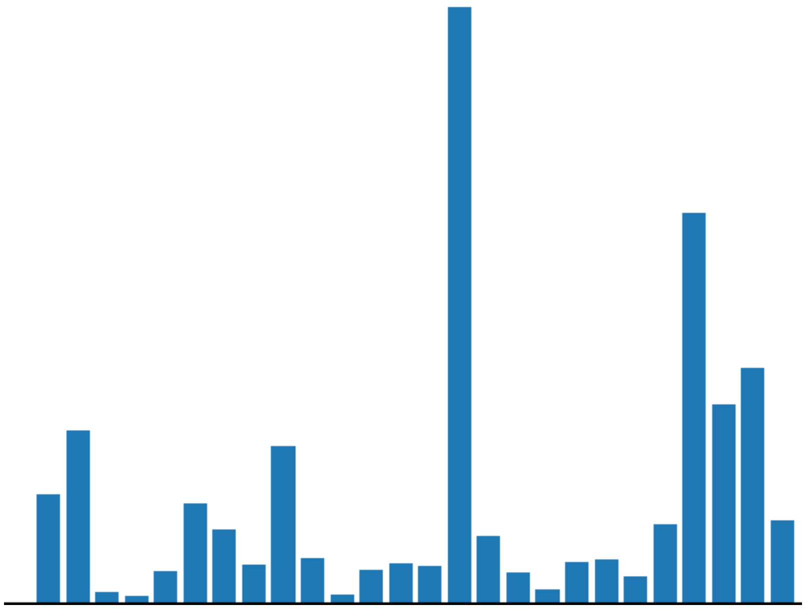


Figure 2.4: Imbalanced Classes Distribution

The choice of keeping just 26 classes was made for simplification purposes, also, it represents the diagnoses of clinical interest, relatively common, and recognisable from ECGs. Those diagnoses are summarised in the table below:

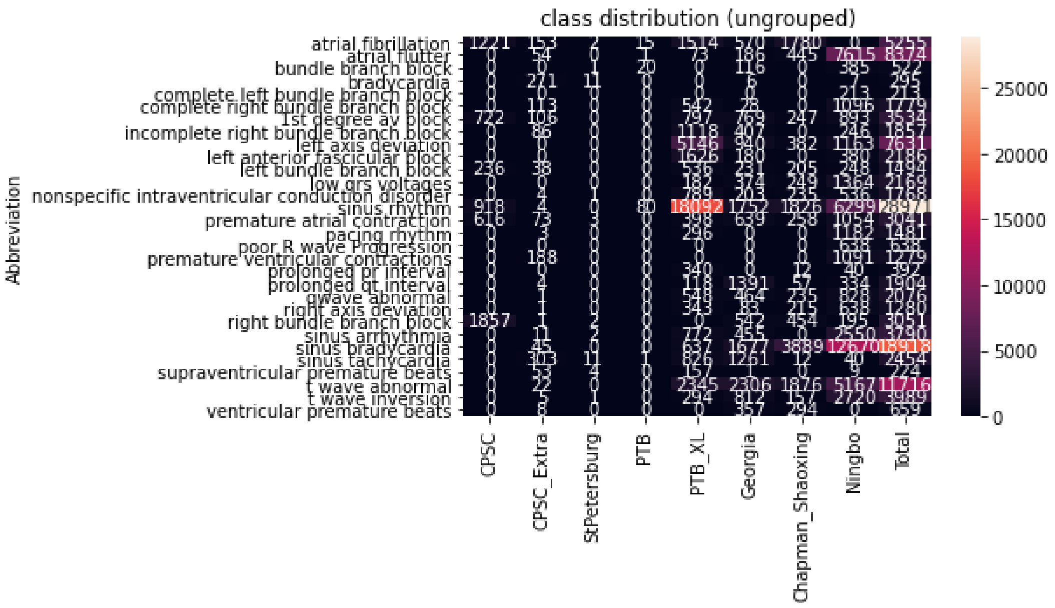


Figure 2.5: The distribution of the 30 classes kept for evaluation

As shown in the table, the databases distribution of abnormalities are generally quite unbalanced, where each database contains a limited number of abnormalities present in their recordings.

2.2 Data structure

Each recording R consists of twelve leads, i.e twelve time series $j \in \{1, ..., 12\}$ that we can model mathematically as the following :

$$R = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix} \tag{2.1}$$

Where $i \in \{2, 3, 4, 6, 12\}$ is the number of leads to be used for prediction.
That means that every step of the prepocessing that we will see in the following section is applied to every chosen leads from the twelve ECG leads.

Chapter 3

Methods

The raw data were sampled from different sources, which vary in sampling rate, signal amplitude, noise level, etc. In order to mitigate the variance of the data for training, the following data prepocessing techniques were adopted.
These prepocessing steps will be applied only to the relevant leads that will be used for evaluation(two, three, six, or all the twelve leads if its the case).

3.1 Prepocessing

The preprocessing methods used in our work are as follows:

- Cropping
- Filtering
- Resampling
- Normalization

Features extraction method will not be used, our preprocess is based only on deep learning techniques.

3.1.1 Truncating padding

In order to Mitigate the variance between different recordings, cropping is applied to all ECG-leads. We crop 10 seconds long segment out of each ECG-leads, this segment is centred in order to obtain a more reliable reading (due to various reasons such as censors stabilisation, patients movement...).
If the ECG recording is shorter than 10 seconds, zero padding is applied on both sides to keep the original recording centred.

3.1.2 Filtering

Most ECG signals present some power-line noise that is characterised by a sinusoidal pattern of high frequency around 50Hz or 60Hz. Two notch filters, 50 and 60Hz are used in order to delete this electrical current noises.

Another type of filtering is applied, a 3rd order Butterworth band-pass filter (0.5-120Hz) was used to treat that. The lower threshold of 0.5 is set in order to delete low basline noise frequencies under 0.5 Hz, while the upper threshold is set so that the ECG recording don't contain frequencies higher than 125 Hz in order to resample it at 250 Hz.

3.1.3 Resampling

Since there was three different sample frequencies in our data set : 257Hz, 500Hz and 1000Hz, in order to unify the recording frequencies , down-sampling was applied, with 250Hz as the sampling frequency.

3.1.4 Normalisation

To avoid large signal amplitudes affecting the prediction, the 12-leads ECGs were normalised with a Z-score normalisation, see the formula below:

$\tilde{z}(t) = \frac{z(t)-\tilde{z}}{\sigma}$

Where:

- $\tilde{z}$ Is the mean calculated over the given ECG R=($x_1, x_2..., x_i$) (2.1) one element at a time.
- σ Is the standard variation of the same ECG (2.1) one element at a time as well.

So after this step all the ECGs have a mean of 0 and a standard deviation of 1. The figure bellow shows an example of an ECG lead after all the preprocessing steps.

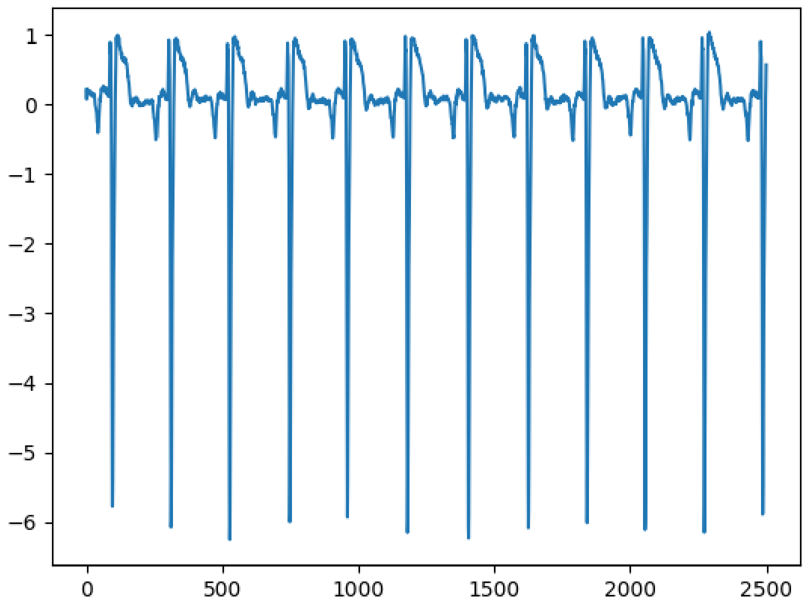


Figure 3.1: ECG lead after preprocessing

3.1.5 Stratified k-fold

In order to ensure that both the training and validation sets are reflective of the data' various classes Stratified k-fold was applied to split the data (typically to 5 sets). This method is very helpful when the data is unbalanced especially if we try to work with one data sat at a time which make things even more unbalanced. This is beneficial since it helps us to handle classes imbalance over the different splits that may contain some information about what we are trying to classify.
The reasoning behind this, is based on the fact that most classification algorithms tend to be biased. They tend to weight each instance uniformly, resulting in over-represented classes receiving an excessive weight (e.g. F-measure, Accuracy or a complementary form of error) which becomes a bigger problem if we trained the model on one data set at a time due to the slight presence or absence of most abnormalities, so if one of the classes that are represented in only few of the recordings and all those recordings end up being in validation set this can heavily impact the model performance.
In our case, we used the stratified 5-fold to ensure that we had 4/5 of the data for training and the remaining 1/5 for validation.

3.1.6 Model input

According to the challenge our model will be assessed on various type of inputs, two leads, three leads, six or the whole 12 ECG leads.
In order to optimise the training time and to create one model for the all of the task, The choice was to feed the model each of the ECG leads separately as different samples (with the same label for the same record), so the vector of ECG leads R(t) (2.1) will be separated into i unique inputs for the model training , in this logic the model will be the same for all the types of input provided by the challenge.
In addition, treating the ECG leads this way serves as somewhat a data augmentation technique which can make the relatively small data set to a considerable one(up to twelve times more data).
In terms of medical view, this approach is proper since most abnormalities can be detected using only one ECG lead.
In the validation/testing steps a combination of the twelve (or less depending on the case) prediction of the model will be used to make the final classification.

3.1.7 Cyclic learning rate

To accelerate our models convergence, we chose to integrate a cyclic learning rate scheduler [14]. A high learning rate causes the model to fluctuate, while a lower learning rate can cause the model to converge very slowly or to converge to a local minima. Cyclical learning rate scheduler allows keeping the learning rate high and low, causing the model not to diverge besides jumping from the local minima or the saddle point.
Cyclical learning rate policy changes the learning rate after every batch.

The policy cycles the learning rate between two boundaries with a constant frequency, it oscillates between base learning rate and max learning rate.

In our model the oscillation of learning rate is based on the triangular window which makes the learning rate changing logic linear that is we will increase the learning rate with some constant from min learning rate to max learning rate and will decrease the learning rate with the same constant from max learning rate to minimum learning rate after each step.

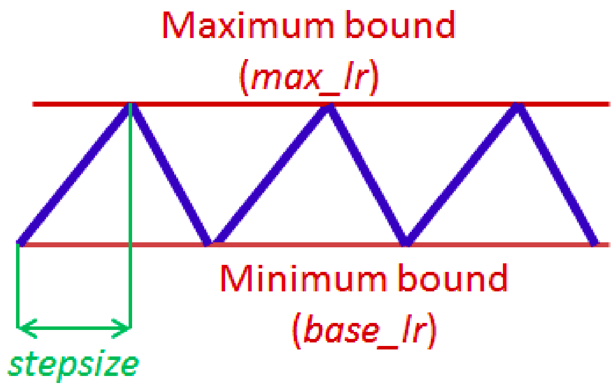


Figure 3.2: Triangular window [15]

the lower and upper bounds of the learning rate were set based upon our previous testing of the models.
The figure bellow show the mean of the learning rate on each epoch while training.

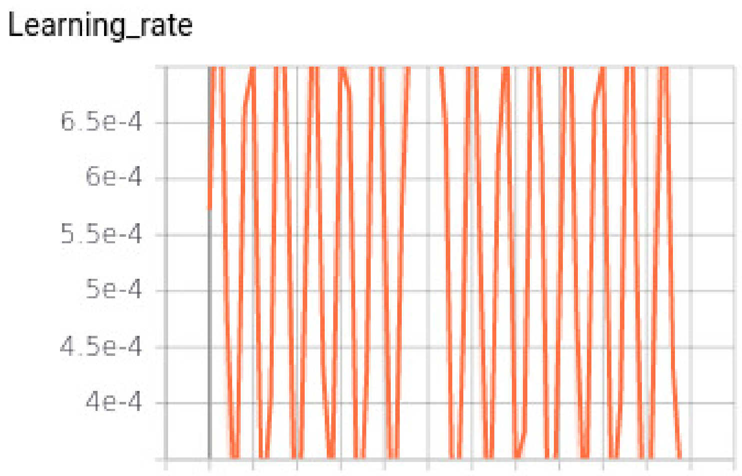


Figure 3.3: : Learning rate variation during training

3.2 Metrics

3.2.1 Losses

For our model a combination of many losses were tested. Below we will have a thorough description of the ones used, and a brief description of those tested.

BCE with logits loss

This loss is a combination of the BCELoss along with a sigmoid a Sigmoid layer in one class. To do so, it uses the log-sum-exp trick which helps getting numerical stability and thus better performance, therefore it performs better than using the BCE after the sigmoid layer of the network. [16] Below are the formulas of this loss :

$l_c(x, y) = Lc = l_{1,c}, l_{2,c}, ...l_{N,c}$, with $lc = -w_{n,c}[p_c y_{n,c}. \log(\sigma(x_{n,c}))+(1-y_{n,c}). \log(1-\sigma(x_{n,c}))]$
where:

- N is the batch size.
- C is the class number.
- p_c Is the weight of the positive answer for the class c.
- σ represents the sigmoid function.

pc is used to trade off recall and precision by adding weights to positive examples.
Precision : The ratio of correctly predicted positive observations to overall predicted positive observations.
Recall : Also called as sensitivity, it is the ratio of correctly expected positive observations to all observations in the actual class.
These metrics are mathematically defined as follow :

Recall $=\frac{TP}{TP+FN}$
Precision $=\frac{TN}{TN+FP}$

$pc > 1$ increases the recall whereas pc < 1 increases the precision. Therefore pc > 1 penalises more a false negative, which makes sense because missing a disease is a big issue whereas diagnosing a health patient can be corrected without dramatic consequences.
Below are the performance on all the data sets obtained by using this loss with the ConvnetSEClassifier model.

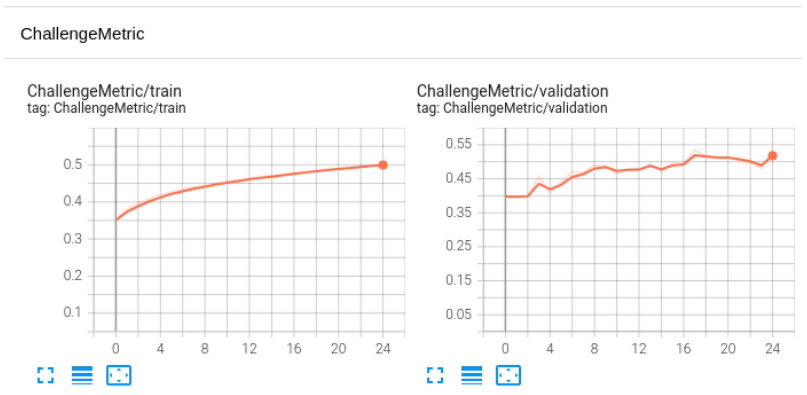


Figure 3.4: Challenge metric using BCEwithLogits

Soft Dice loss

Another common loss for this tasks is the Dice similarity coefficient, which is a measure of how well two contours overlap [17]. The dice loss is usually applied in segmentation tasks as in 3D medical image segmentation, in our case we will apply this similarity measure on the labels array.

The Dice index ranges from 0 (complete mismatch) To 1 (perfect match). In general, for two sets A and B , the Dice similarity coefficient is defined as:

DSC(A,B)=$\frac{2*|A \cap B|}{|A|+|B|}$.

Here we can interpret A and B as sets of voxels, A being the predicted abnormalities present and B being the ground truth.

Where the model will map each voxel to 0 or 1:

- 0 means the anomaly is not present
- 1 means the anomaly is present

The dice coefficient "DSC" is:

DSC(f,x,y)=$\frac{2*\sum_{i=1}^{c} f(x)_i y_i+\epsilon}{\sum_{i=1}^{c} f(x)_i+\sum_{i=1}^{N} y_i+\epsilon}$

In the dice coefficient, the variables in the formula are:

- x : the input Time series
- f(x) : the model output (prediction)
- y : the label (actual ground truth)
- is a small number that is added to avoid division by zero(usually set to one).

And The Dice loss formula is :

$L_{Dice}(f, x, y) = 1–DSC(f, x, y)$

Yet, even though the dice loss makes a better intuitive sense , we will be using the soft dice loss which is much better for training (its harder to back backpropagate due to discrete values), the only difference being that instead of zeros and ones we'll be using the model outputted probabilities. the Soft Dice loss formula is:

$$L_{Dice}(p, q) = 1–\frac{2*\sum_{i=1}^{c} p_i*q_i+\epsilon}{\sum_{i=1}^{c} p_i^2+\sum_{i=1}^{c} q_i^2+\epsilon}$$

- p is our predictions
- q is the ground truth
- In practice each qi will either be 0 or 1.

- is a small number that is added to avoid division by zero

The main objective behind using this loss is to push the model in focusing further more on obtaining a high accuracy.

DiceBCE loss

This loss is the sum of the two losses defined above, see the formula below:
$L_{DiceBCE}(p, q) = L_{Dice}(p, q) + L_{BCEWithLogits}(p, q)$

This loss had the best performance overall since it was able to handle the classes weights and imbalance while maintaining a high accuracy.
Below are the performance on all the data sets obtained by using this loss with the ConvnetSEClassifier model.

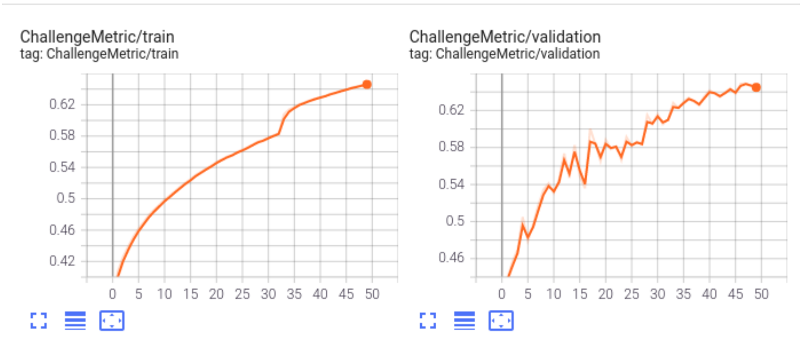


Figure3.5: Challenge metric using DiceBCE loss

3.2.2 Other Losses and performance comparison

Another loss that we considered to handle class imbalance was the SignLoss, it was implemented by the third team on the leader-board in the physionet 2020 challenge [18] and like the previous case with the BCEwithLogits, we added this loss with the dice loss to focus more on the accuracy.
The formula of this loss is :

$$\begin{cases} y - 2py + p^2, & \text{if } |p - y| < 0.5 \\ 1, & \text{if } |p - y| \geq 0.5 \end{cases}$$
$L_{Sign} = \sum_{i=1}^{c} Sign(p_i) L_{DiceSign} = L_{Sign} + L_{Dice}$

The performance obtained using the Sign loss alone and the DiceSign loss is shown below:
As shown in the figure, the idea of mixing these two losses was not a success.

Finally, the comparison between the different losses performance is shown below:

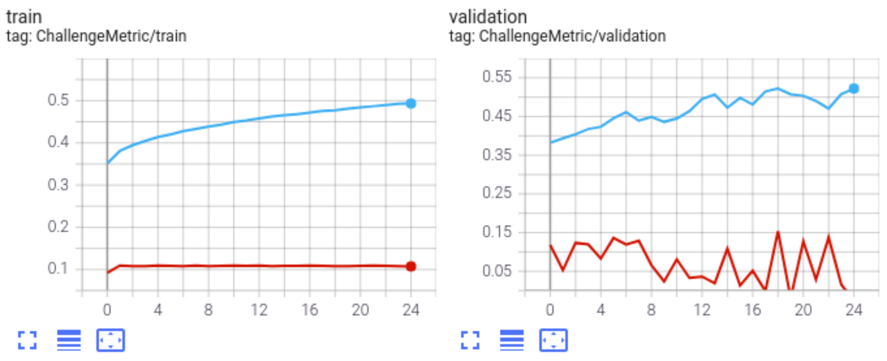


Figure 3.1: Challenge metric using DiceSign and Sign loss

tensorboards_ampere/se_dicesign_0
tensorboards_ampere/se_sign_0

3.2.3 Performance Metrics

Accuracy

The most basic performance metric is accuracy, which is essentially the ratio of correctly expected observations to all observations. A high accuracy does not necessarily imply that our model is excellent, yet the challenge computed accuracy is not so intuitive as counting the correctly classified abnormalities and dividing by the total, rather it is computed on the whole ECG 26 abnormalities, so an example is considered correct only if its was correctly classified on the whole abnormalities. In this case accuracy can be a good metric to evaluate our model overall performance, yet we have to look for further metrics to assess our model's performances.

Accuracy = (T P + T N)/(T P + T N + F P + F N)

Challenge metric

In order to create a metric reflective of the real world healthcare system, the used metric in the 2021 challenge gives partial praise to some misdiagnoses that are somewhat similar to the correct diagnoses (both abnormalities have an identical health effect or are similarly treated by the cardiologists) identical effects or procedures as the true diagnoses, as determined by the cardiologists.

This measure illustrates the scientific fact that it is less harmful to confuse some abnormalities than others because the responses may be similar or the same and that some misdiagnoses are more dangerous than others, and they should be graded accordingly.

Let C = $[c_i]$ be the list of the different diagnostics.The multi-class confusion matrix is defined as follows:
A = $[a_{ij}]$, where a_{ij} is the number of recordings listed as belonging to class c_i but currently belong to class c_j in a database (c_j and c_i may be the same class or an

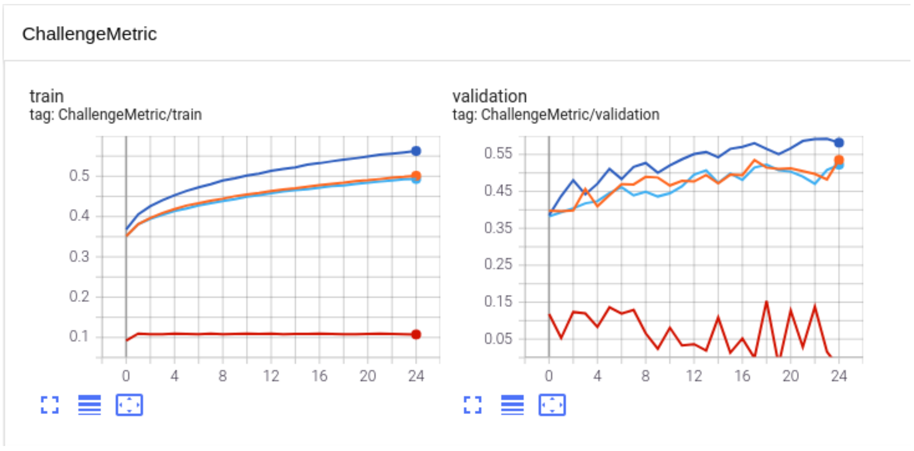
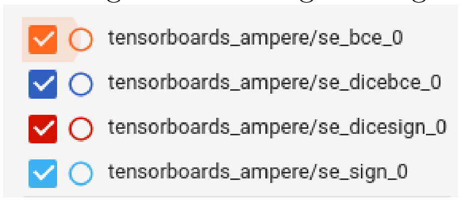


Figure 3.2: Challenge metric using DiceSign and Sign loss


✓ ○ tensorboards_ampere/se_bce_0
✓ ○ tensorboards_ampere/se_dicebce_0
✓ ○ tensorboards_ampere/se_dicesign_0
✓ ○ tensorboards_ampere/se_sign_0

equivalent classes). Based on the similarities of treatments or risk variations, we apply different weights W = $[w_i j\ ]$ to different entries in this matrix.

s=$\sum^m \sum^m{}_{i=1}\ {}_{j=1}\ w_{ij} a_{ij}$
is a simplified version of the standard scoring metric that yields the score s. The value s is then normalised with a score of 1 for a classifier that always outputs the true classes and a score of 0 for an ineffective classifier that always outputs the standard class.
For more details about the challenge metric view the challenge [6]

F1 score

Since we are dealing with a multi-label classification problem with 26 unbalanced classes, and that some classes dominate the others throughout the whole data sets, even if the way the accuracy is defined makes it a good way to assess the model performance, it lacks the specific information about each class separately.In order to observe the model performance on each class, we calculated the F_1 score on each class separately and the overall F_1 score which takes into account false positives and false negatives for each class. To define the F1 score, we need the precision and recall as defined earlier:

$$F_1 = \frac{2*Precision*Recall}{Precision+Recall}$$

3.3 Tested architectures

Our approach for developing the models was classic, we starts with a simple network of the model we chose then complexifying it till it overfits the training data and after that we start introducing regularisation steps.

Knowing that deeper networks generally perform better in such tasks, the strategy was to add more layers and playing with all the parameters of each layer filters. Once the network overfits we add regularisation with dropout layers, L2-regularisation, weight decay...

For our project, we used NVIDIA GeForce GTX 1080 Ti.

3.3.1 Convnet-SE

This architecture was inspired from the model created by the third team on the leader board for last year challenge [18], our base model was another model (OldNoaraConvnet) in which we stacked Squeeze-excitation blocks [19] and an inception module integrated within to enhance the gradient propagation .

Due to the large class imbalance present in our data in which some classes are present with only few samples, such imbalance could undermine the model's performance, as the model is likely to learn the pattern from categories with a large number of samples while ignoring the minorities which could be the more important abnormalities to detect.

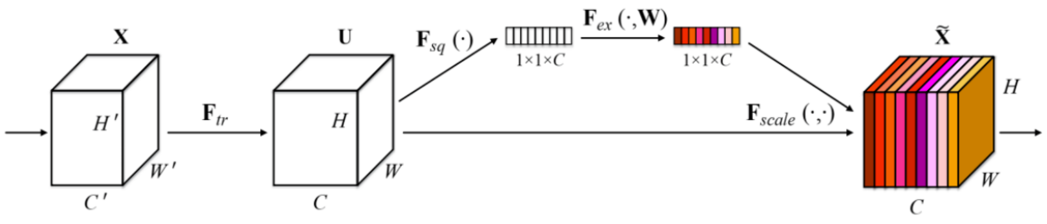


Figure 3.6: Squeeze and excitation block [20]

The main idea is to focus on the Chanel wise relationship which is the correlation between the different channels (the depth information) instead of the temporal one (learned by the kernel while scanning on the whole sequence), so we allow the network to perform feature re-calibration in which we suppress the less use full features and emphasise on the important ones .
The squeeze operation is a simple global average pooling which compresses the temporal information to a single dimension and produces an embedding of the global distribution of feature responses for each channel, so the originally spread out temporal information gets squeezed to a single number for each channel, in this way, the information from the global receptive field can be used by all the following layers of the network.

The excitation operation takes the learned embedding by the squeeze operation as input and captures the channel-wise dependencies, then produces weights for each

channel in the network. Consequently, the learned scaled weights are applied to previously learned feature maps to realise the feature re-calibration.
In this way, we give another degree of freedom to the network so that it can learn which channel is more important for the classification task at hand.

The final SEInception layer consist from a stack of :

- Average pooling (Squeeze operation).
- Two linear layers with a Relu activation function (the excitation operation), in which the reduction factor is set to 16.

The figure bellow gives an overview of the block:

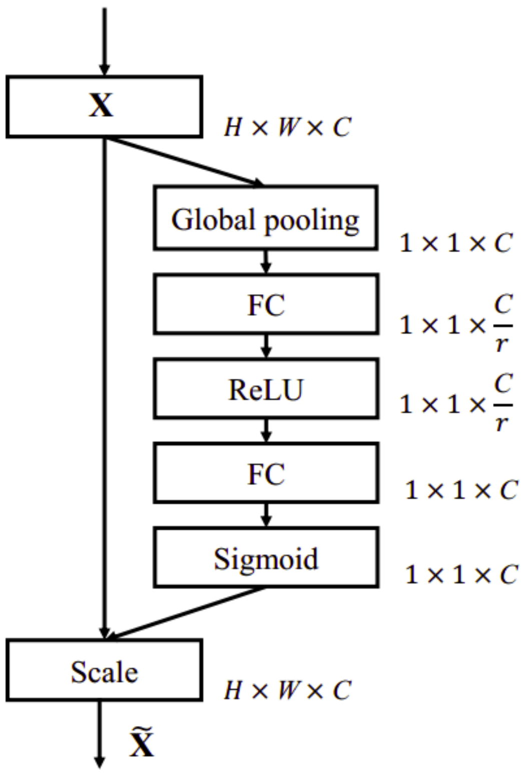


Figure 3.7 Squeeze-and-Excitation block [21]

Architecture setup

Our model consists of 16 blocks, where the output channel size varies from 16 at the first layer to 512 at the final layer,this size gets multiplied by two after every three layers, each block consists in a 1D convolutional operation with the kernel being equal to eleven followed by a batch normalisation and then the output is passed through a Relu activation function, these blocks are the base model (OldNoraConvnet) to which we then added the SEInception blocks at the middle and later blocks so that the excitation is more class-specific, finally we pass the output to a global max pooling and then to a small neural network to make the final classification.

The global setup is as follows:

- Batch size: 60
- Optimizer : Stochastic gradient descent
- learning scheduler :Cyclic

The figure bellow shows an overview of the model:

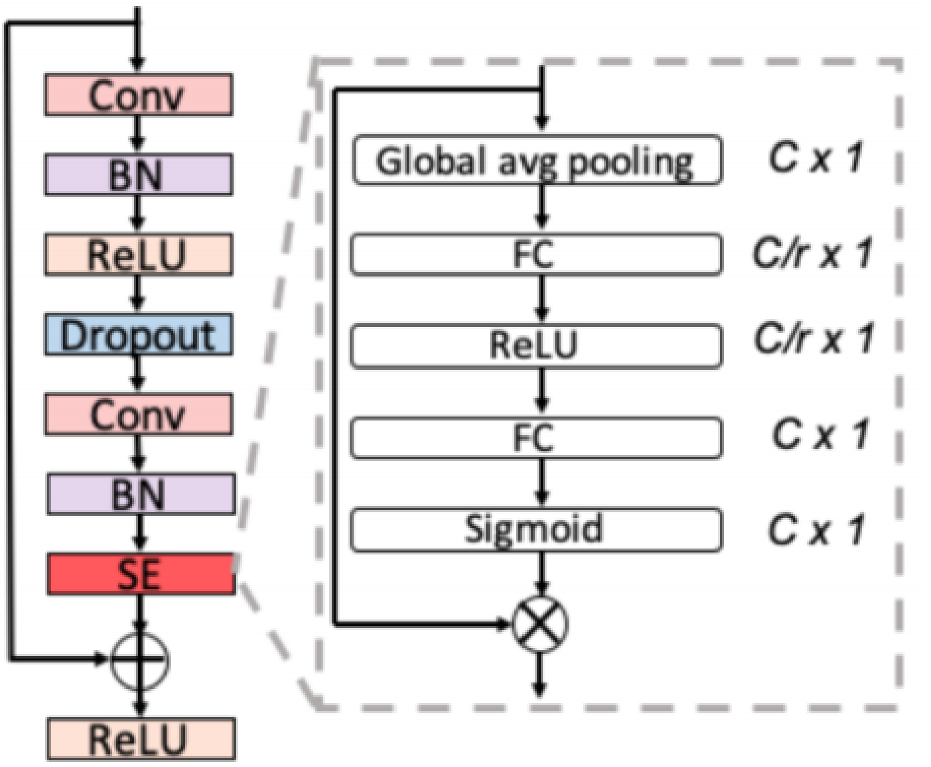


Figure 3.8: Convnet-SE model

3.3.2 ResNet

This architecture was developed by entralesupelec students for the last year challenge version, they created some ResNets (inspired from Pytorch) [22] with different number of layers : 18, 34, 50, 101 that were adapted for 1D inputs.

To keep it memory efficient, the residual block is different when there is a large number of layers (50,101...) It consists in computing a 1*1 convolution to reduce the shape of input channels before applying the 1*3 convolution and then applying a 1*1 convolution to retrieve the original shape. This allows to consume less memory than the traditional 1*3 convolution in the basic residual block of Resnet18 Resent34

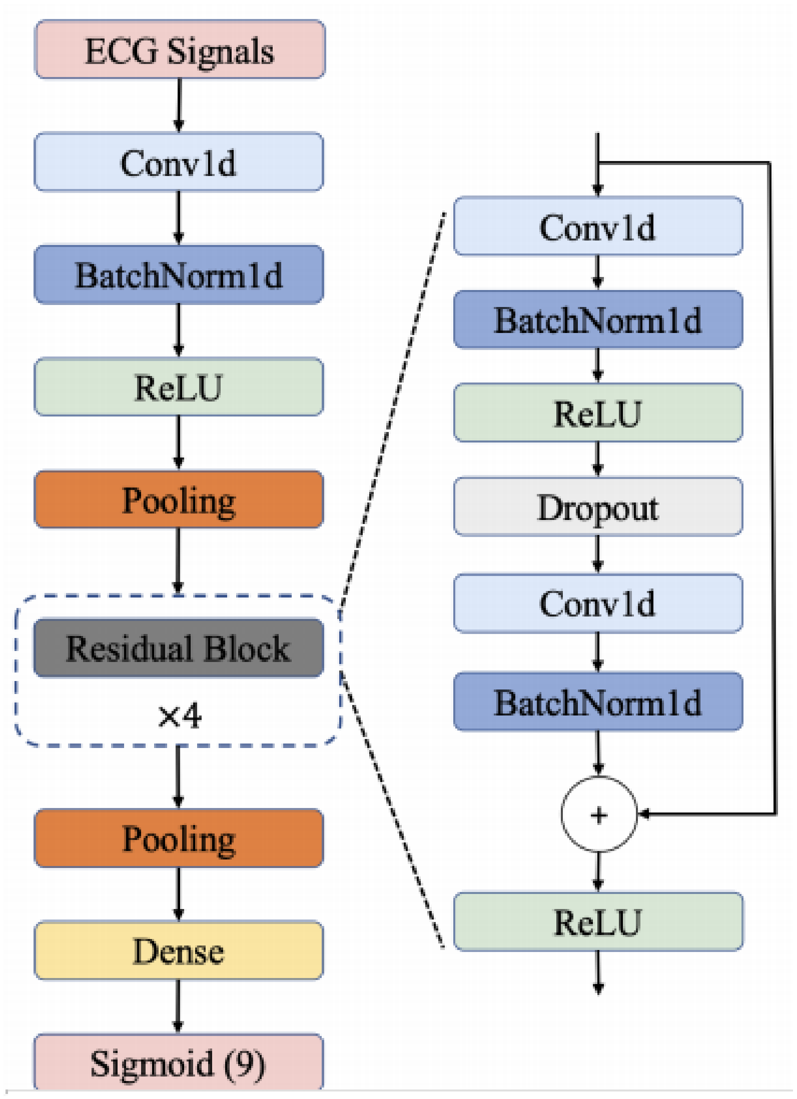


Figure 3.9: Resnet34 architecture [23]

3.3.3 Transformer

The Transformer is an architecture that uses attention mechanism to boost the speed with which the model can be trained by using parallelization unlike typical recurrent neural networks that require sequential computation.
The Transformer was proposed in the paper Attention is all you need [24].

Since this is a classification problem, we only used a stack of encoders to define our attention mechanism between the different extracted features, so the basic components of our model will be as follows:

- Embedding layer
- Positional encoding layer
- Multi-head Attention layers

Embedding

For the embedding layer, since the predefined layer by pytorch is specific for an NLP tasks and is not compatible with our type of input, the official documentation uses word embedding to create an embedding matrix of a fixed vocabulary size, the same technique can't be applied to model variable features. In order to define our own embedding, we chose to apply a series of convolutions to encode the deep features in the input and learn an embedding of the raw ECG waveform.
To make sure that this way of defining the embedding layer is efficient and that this layer is able to extract enough features for our transformer model, we chose to integrate a convolutional model that is able by itself to make the classification task with a good final metric.
The convolutional models that were tested are first the Resnet models built for the last year challenge version by Centralesupelec students, also the SEconvnet previously defined model was tested for the embedding task.

Positional encoding

For the positional encoding, we used the predefined class by pytorch [11], positional encoding module is used to injects information about the relative or absolute position of the extracted features in the ECG time serie. The positional encoding have the same dimension as the embedding (the number of the extracted features by the convolutional layers) so that the two can be summed. Sine and cosine functions of different frequencies are used to obtain this positional encoding, the calculation steps are as follows:
Let t be the desired position in an input sentence, $\vec{P_t} \in \mathbb{R}^d$ be its corresponding encoding, and d be the encoding dimension . Then, $f : \mathbb{N} \to \mathbb{R}^d$will be the function that produces the output vector $\vec{P_t}$ and it is defined as follows:

$$\vec{P_t}^{(i)} = \begin{cases} \sin(w_k.t) & \text{if } i = 2k \\ \cos(w_k.t) & \text{if } i = 2k+1 \end{cases}$$

where:
$w_k = \frac{1}{10000^{2k}}$

and the final vector $\vec{P_t}$:

$$\vec{P_t} = \begin{bmatrix} \sin(\mathrm{w}_1.t) \\ \cos(\mathrm{w}_1.t) \\ \vdots \\ \sin(\mathrm{w}_{(d/2)}.t) \\ \cos(\mathrm{w}_{(d/2)}.t) \end{bmatrix} \tag{3.1}$$

The frequencies are decreasing along the vector dimension. Thus it forms a geometric progression from 2π to 10000.2π on the wavelengths.

The positional encoding is then added on top of the actual embedding. That is for every extracted feature from the ECG lead. So the final output which is then fed to the model is calculated as follows:
$I_t = F_t + \vec{P_t}$

To make this summation possible, the positional embedding dimension must be equal to the word embedding dimension i.e.

$d_{embedding} = d_{posionalEncoding}$

This encoding is not integrated into the model itself, this vector is used to equip each feature with information about its position in the time serie. In other words, we enhance the model's input to inject the order of the features.

Encoder layers

Learned embedding that are summed with the positional encoding are then fed into a Transformer architecture, which relies entirely on a parallelizable self attention mechanism.
Our attention layers consist only of a stack of encoders, The encoders are all identical in structure, each one is broken down into two sub-layers:

- A self-attention layer, its a layer that enables the encoder look at other features in the sequence as it encodes a certain feature.
- A feed-forward neural network. The exact same feed-forward network is independently applied to each position.

There are also residual connections around each two encoders sub-layers followed by a layer normalisation.

The multi-headed attention output vector is added to the original positional input embedding. This is called a residual connection. The output of the residual connection goes through a layer normalisation. The normalised residual output gets projected through a point-wise feed-forward network for further processing. The point-wise feed-forward network is a couple of linear layers with a ReLU activation in between. The output of that is then again added to the input of the point-wise feed-forward network and further normalised.

The residual connections help the network train, by allowing gradients to flow through the networks directly. The layer normalisation's are used to stabilise the network which results in substantially reducing the training time necessary. The point-wise feed-forward layer is used to project the attention outputs potentially giving it a richer representation.

N times encoders can be stack to further encode the information, where each layer has the opportunity to learn different attention representations therefore potentially boosting the predictive power of the transformer network.

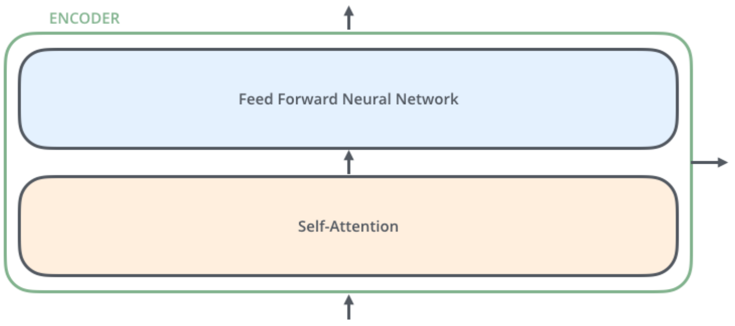


Figure 3.10: Encoder structure [25]

After embedding each feature in the input sequence, each of them flows through the two layers of the encoder. One key property in the transformer is that the various features in each position flows through its own path in the encoder. The dependencies between these paths are solely in the self-attention layer, the feed-forward layer is unrelated to those dependencies, with this logic the various paths can be executed in parallel while flowing through the feed-forward layer which constitutes the key upper hand the transformer architecture have on RNNs.
In order to make this a multi-headed attention computation, the query, key, and value which are the input representation by the model, gets split into N vectors before applying self attention process. The split vectors then go through the self-attention process individually. Each self attention process is called a head. Each head produces an output vector, those vectors gets concatenated into a single vector before going through the final linear layer.
Theoretically, each head would learn a different relation therefore giving the encoder model more representation power.

The output of the final encoder layer goes through a final linear layer, that acts as a classifier.

The figure below gives an overview of the complete system:

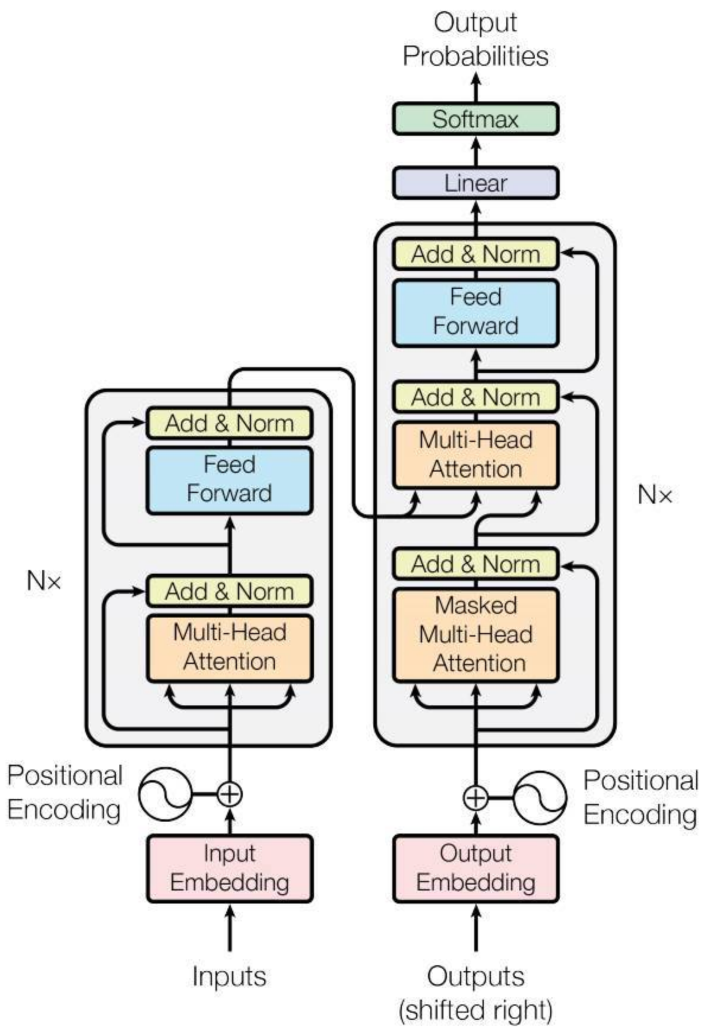


Figure 3.11: Overview of the transformer architecture [11]

Architecture setup

The global model setup is as follows:

- Number of encoder stacks : 16
- Number of heads : 8
- Linear layer dimension : 2048
- Batch size: 60
- Optimizer : Stochastic gradient descent
- learning scheduler :Cyclic

The final classifier consist of three linear layers with a LeakyRelu as the activation function, in the result section the embedding layer used will be the ConvnetSE model sec:convnetSE .

Chapter 4

Results

4.1 Challenge results

The final challenge result won't be revealed till mid September, the feedback we have for now was only for our early version of the convnetSEClassifier sec:convnetSE obtaining scores of 0.587, 0.574, 0.574, 0.577 and 0.572 on 12-lead, 6-lead, 4-lead, 3-lead, 2-lead validation data sets were obtained respectively for the challenge's metric.

	12-leads	6-leads	4-leads	3-leads	2-leads
Challenge metric	0.587	0.574	0.574	0.577	0.572

The transformer model was not submitted due to its longer execution time that exceeds the challenge restrictions, yet two other versions of the convnetSEClassifier were submitted in waiting for the final result.
Currently the best team submission obtains scores of 0.72, 0.68, 0.70, 0.70, 0.68 on 12-lead, 6-lead, 4-lead, 3-lead, 2-lead validation data sets respectively.

The comparison between our different models will be based on our local environment, and due to GPU Memory Restriction on the cluster at CentraleSupélec, the performance comparison will be based mainly on the Ningbo data set since we can't execute the code on the whole data set. Ningbo is the biggest and most representative data set of the training data, it contains 28 out of the 30 scored abnormalities and about %50 of the data.

4.1.1 Comparing Models

First, we tested the various models implemented on the Ningbo data set. Bellow, we show the different metrics on the validation and training set for each model trained on 55 epochs and with the whole 12 leads as input.
The two compared models are the SEconvnetClassifier and the transformer model, we neglected the resnet since it has given a far weaker performance results than the

other two since the begining.

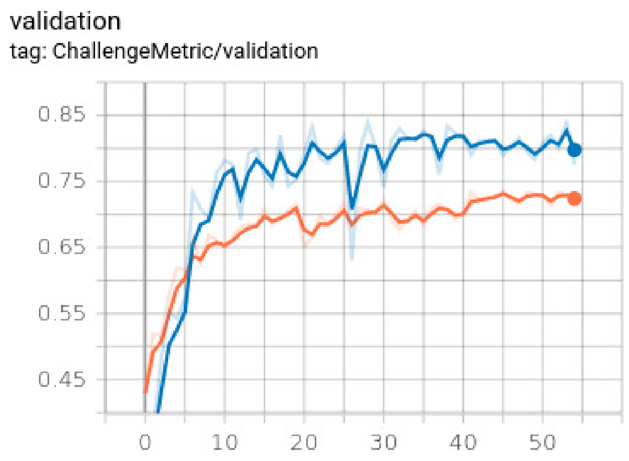


Figure 4.1: Challenge metric on the Ningbo data set, using ConvnetSEClassifier and SETransformer

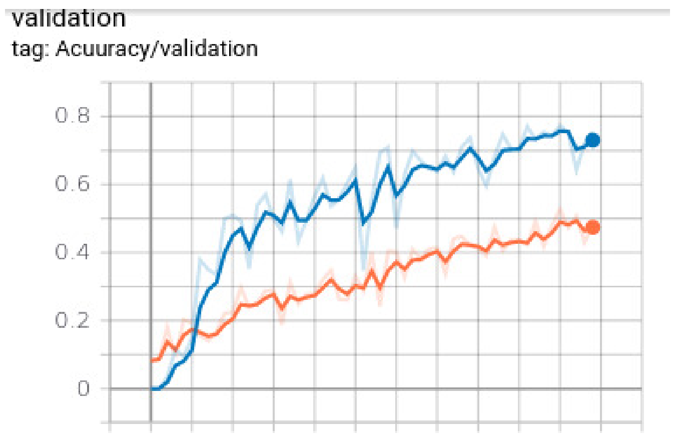


Figure 4.2: Accuracy on the Ningbo data set, using ConvnetSEClassifier and SETransformer

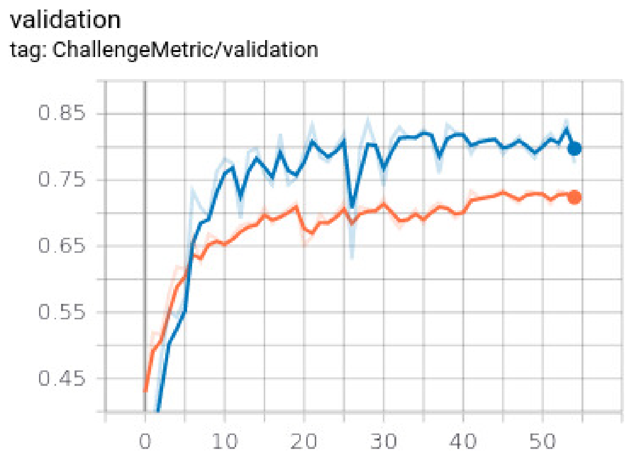
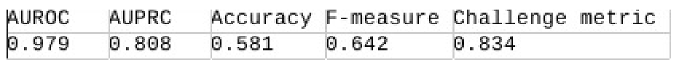


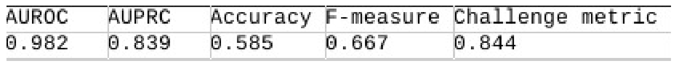
Figure 4.3: Loss on the Ningbo data set, using ConvnetSEClassifier and SETransformer

And bellow are the evaluation metrics calculated in the challenge using the whole Ningbo data set for testing :



AUROC	AUPRC	Accuracy	F-measure	Challenge metric
0.979	0.808	0.581	0.642	0.834

Figure 4.4: Challenge evaluation metrics for the ConvnetSEClassifier



AUROC	AUPRC	Accuracy	F-measure	Challenge metric
0.982	0.839	0.585	0.667	0.844

Figure 4.5: Challenge evaluation metrics for the SEtransformer

The best model is the SEtransformer according to each metrics. However , given that each data set have its specificity from the equipment used(influence directly the features present in ECG lead) or the frequencies of each abnormalities, more testing should be done to make sure which model is able to keep a high score while mitigating each data set specificity and maintain a balance between them to generalise better .

4.1.2 Best model's results

For the following results, we used SEtransformer on all of the 12 leads.

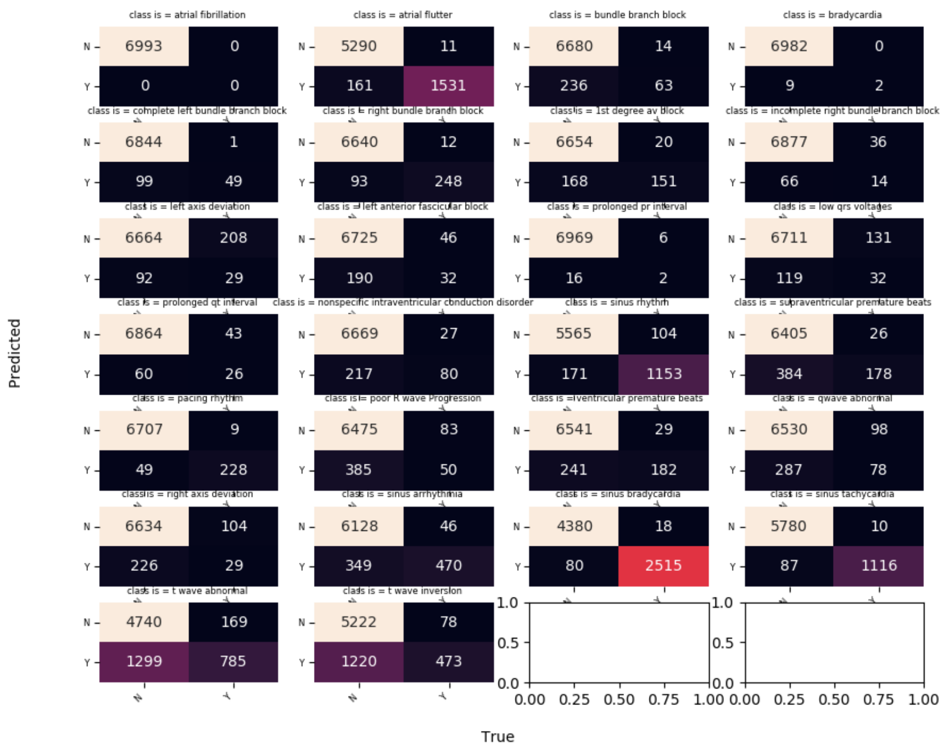


Figure 4.6: Confusion matrix on the validation set

We plotted above the confusion matrix (for each of our 26 classes) after testing our model on the validation set that contains 6993 elements. In our graph, "Y" means true and "N" is for false.

Below, the different metrics per class on the whole Ningbo data set for the SEtransformer are shown, if the value is "nan" that means that the class is not present in the data set:

Classes	AUROC	AUPRC	F-measure
164889003	nan	nan	nan
164890007	0.999	0.998	0.965
6374002	0.993	0.810	0.432
426627000	1.000	0.794	0.560,
164909002|733534002	0.999	0.975	0.829
59118001|713427006	0.999	0.989	0.889
270492004	0.996	0.870	0.710
713426002	0.968	0.834	0.549
445118002	0.968	0.368	0.440
164947007	0.938	0.778	0.420
251146004	0.978	0.788	0.441
111975006	0.990	0.826	0.587
698252002	0.966	0.853	0.671
426783006	0.983	0.824	0.594
284470004|63593006	0.993	0.974	,0.919
10370003	0.994	0.923	0.731
39732003	0.994	0.991	0.944
365413008	0.999	0.690	0.423
17338001|427172004	0.971	0.945	,0.753
164917005	0.994	0.761	0.454
47665007	0.972	0.813	0.510
427393009	,0.963	0.932	0.805
426177001	0.993	0.997,	0.983
427084000	0.915	0.996,	0.971
164934002	0.965	0.556	0.583
59931005	0.963	0.682	0.520

Chapter 5

Conclusion

5.1 Acknowledgements

First, I would like to express my sincere gratitude to my all supervisors , Jeremy Fix and Michel Barret from CentraleSupélec and from the National Institute of Health and Medical Research (Inserm) Julien Oster and Pierre Aublin.
I'am very grateful for their guidance all along the project and their genuine interest and investment in our project and for the many things they enabled me to learn throughout this internship.

This internship was my first extensive Deep Learning project and a great experience to practice what I have learned throughout this year at Ensta Paris and also to single out the many aspects I need to focus on in order to improve. I also would like to thank CentraleSupélec for providing me with a great hardware during the whole internship.

5.2 Conclusion and Discussion

To conclude our work, we created a pipeline to classify ECGs based on Deep Learning method using various data sets provided by the Physionet Challenge. We worked mainly on 2 models : The ConvnetSEClassifier sec:convnetSE and the SETransformer sec:SETransformer . After creating the models, We tested different preprocessing methods by changing the filters and trying to give different weights to the abnormalities depending on the data set it came from. The best results locally on the Ningbo dataset were obtained using a SETransformer sec:SETransformer model with 12 leads ECGs as input on 55 epochs.
While observing the challenge test results sec:Challenge results we can validate our previous assumption that most of the abnormalities can be correctly diagnosed using only two ECG leads and that the use of the whole 12 leads gives only a slight improvement. To improve and complete our work, we suggest several ideas :

- Test different optimizers
- Integrate some useful static features as the sex and age of a patient.

- Give different weights to the various predictions given the lead it was predicted from, since the features for some abnormalities varies from one lead to another.
- Optimise the preprocessing time so that the models take less time for training.

Bibliography

[1] “World Health Organization. Cardiovascular diseases (CVDs). 2017. url:.” https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[2] “ Zahra Ebrahimi , Mohammad Loni , Masoud Daneshtalab , Arash Gharehbaghi: A review on deep learning methods for ECG arrhythmia classification url:.” https://www.sciencedirect.com/science/article/pii/S2590188520300123.

[3] “Zhaohan Xiong 1, Martyn P Nash, Elizabeth Cheng, Vadim V Fedorov, Martin K Stiles, Jichao Zhao : ECG signal classification for the detection of cardiac arrhythmias using a convolutional recurrent neural network url:.” https://pubmed.ncbi.nlm.nih.gov/30102248/.

[4] “Philip A Warrick 1 , Masun Nabhan Homsi: Ensembling convolutional and long short-term memory networks for electrocardiogram arrhythmia detection url:.” https://pubmed.ncbi.nlm.nih.gov/30010088/.

[5] “Erick A. Perez Alday1 , Annie Gu , Amit Shah , Chad Robichaux , An-Kwok Ian Wong , Chengyu Liu , Feifei Liu , Ali Bahrami Rad , Andoni Elola , Salman Seyedi , Qiao Li , Ashish Sharma , Gari D. Clifford, , Matthew A. Reyna1 : Classification of 12-lead ECGs: the PhysioNet/ Computing in Cardiology Challenge. url:.” https://physionetchallenges.org/2020/papers/2020ChallengePaper.pdf.

[6] “Will Two Do? Varying Dimensions in Electrocardiography: The PhysioNet/Computing in Cardiology Challenge 2021 url:.” https://physionetchallenges.org/2021/.

[7] “PhysioNet: Brief Introduction url:.” https://physionet.org/about/.

[8] “Paul Kligfield, MD, FAHA, FACC; Leonard S. Gettes, MD, FAHA, FACC; James J. Bailey, MD; Rory Childers, MD; Barbara J. Deal, MD, FACC; E. William Hancock, MD, FACC; Gerard van Herpen, MD, PhD; Jan A. Kors, PhD; Peter Macfarlane, DSc; David M. Mirvis, MD, FAHA; Olle Pahlm, MD, PhD; Pentti Rautaharju, MD, PhD; Galen S. Wagner, MD : Recommendations for the Standardization and Interpretation of the Electrocardiogram url:.” https://core.ac.uk/download/pdf/81940318.pdf.

[9] "ECG Interpretation :charachteristics of the normal ECG (p-wave,QRS complex, ST segment , T-wave url:." https://ecgwaves.com/topic/ecg-normal-p-wave-qrs-complex-st-segment-t-wave-j-point.

[10] " Annamalai Natarajan, Yale Chang, Sara Mariani, Asif Rahman, Gregory Boverman, Shruti Vij and Jonathan Rubin Philips Research North America, Cambridge, United States :A Wide and Deep Transformer Neural Network for 12-Lead ECG Classification url:." https://physionetchallenges.org/2020/papers/107.pdf.

[11] "LANGUAGE MODELING WITH NN.TRANSFORMER AND TORCHTEXT , Pytorch documentation url:." https://pytorch.org/tutorials/beginner/transformer$_tutorial.html.$

[12] " Flambé documentation url:." https://flambe.ai/en/latest/autoapi/flambe/optim/noam/.

[13] "George Lawton :Transformer neural networks are shaking up AI , published in TechTarget url:." https://searchenterpriseai.techtarget.com/feature/Transformer-neural-networks-are-shaking-up-AI.

[14] "Cyclic LR pytorch documentation url:." https://pytorch.org/docs/stable/generated/torch.optim.lr$_{sche}$

[15] "Sanket Doshi : Cyclical Learning Rates , Published in Analytics Vidhya url:." https://medium.com/analytics-vidhya/cyclical-learning-rates-a922a60e8c04.

[16] "BCEWithLogitsLoss Pytorch documentation. url:." `https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html`.

[17] "Shuchen Du : Understanding Dice Loss for Crisp Boundary Detection , Published in AI Salon url:." https://medium.com/ai-salon/understanding-dice-loss-for-crisp-boundary-detection-bb30c2e5f62b.

[18] "Zhaowei Zhu; Han Wang; Tingting Zhao; Yangming Guo; Zhuoyang Xu; Zhuo Liu; Siqi Liu;Xiang Lan;Xingzhi Sun; Mengling Feng. Classification of Cardiac Abnormalities From ECG Signals Using SE-ResNet. 2020 url:." https://arxiv.org/abs/2101.03895.

[19] "Diganta Misra :Channel Attention and Squeeze-and-Excitation Networks (SENet), Publiched in Paperspace url:." https://blog.paperspace.com/channel-attention-squeeze-and-excitation-networks/.

[20] " Paul-Louis Pröve :Squeeze-and-Excitation Networks,Published in Towards Data Science url:." https://towardsdatascience.com/squeeze-and-excitation-networks-9ef5e71eacd7.

[21] "Varshaneya v. , Balasubramanian S, Darshan Gera :RES-SE-NET: Boosting Performance of Resnets by Enhancing Bridge-connections url:." https://www.researchgate.net/figure/Squeeze-and-Excitation-block-adapted-from-2$_{fig}2_3$31195671.

[22] "Resnet Pytorch git. url:." https://github.com/pytorch/vision/blob/master/torchvision/models/resnet

[23] “Resnet Graph git. url:.” https://github.com/onlyzdd/ecg-diagnosis.

[24] “Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin : Attention Is All You Need. 2017 url:.” https://arxiv.org/abs/1706.03762.

[25] “Jay Alammar: The Illustrated Transformer [Blog post] url:.” https://jalammar.github.io/illustrated-transformer/.