INFO8010: Deep Learning Project: A transformer approach for high performance arrhytmia detection on human electrocardiograms

François Lievens,[1] **Andreas Duquenne**,[2] and **Benoit Thielen**[3]

[1] *flievens@student.uliege.be (s103816)*
[2] *andreas.duquenne@student.uliege.be (s163942)*
[3] *benoit.thielen@student.uliege.be (s152907)*

Preliminary note :
All discussed implementation are available in the following git-hub repository: https://github.com/francoislievens/INFO8010-Deep_Learning_Project

I. INTRODUCTION

Electrocardiograms (ECG) are graphs representing the evolution of cardiac electrical activity over time. This electrical activity consists in the measurement of the potential difference (of the order of mV) between two electrodes placed on the patient's skin.

In order to obtain a three-dimensional view of the electrical activity of the heart, a large number of electrodes can be placed on the patient: a number strategically placed on the chest, both arms, and the legs. During a standard cardiological examination in hospitals, 12-lead ECGs are performed. When using portable devices such as Holters, it is possible to use only two electrodes, thus recording a single-lead ECG.

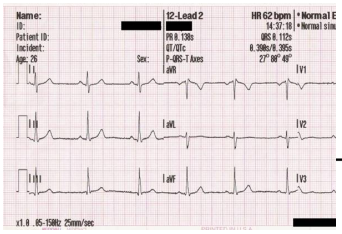

FIG. 1: Example of 4 leads ECG

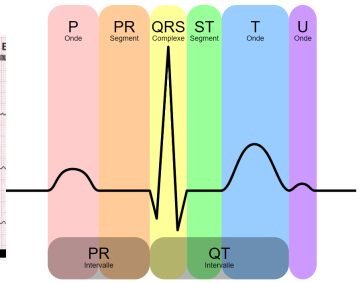

FIG. 2: Classical heart revolution

Without going into details, a cardiac cycle will classically give a wave with the shape of the figure 2 .This wave can then be decomposed into P waves (atrial depolarization), a PR interval (time taken by the electrical signal to pass from the atria to the ventricles), a QRS complex (depolarization of the ventricles and blood ejection), and the T wave (repolarization of the ventricles).

For many years, computer-assisted analysis of ECG traces has been developing with several objectives:

- Measuring certain key parameters accurately and automatically, such as heart rate
- Improve the quality of ECG signals, which may be of very poor quality due to bad sensors or interference. [1].
- Automating the analysis of long-term electrocardiograms
- Improving the detection rate of cardiac abnormalities which can sometimes be difficult to spot by even a highly trained cardiologist. [2]
- Sometimes deep learning can even detect patterns so subtle that they are unknown in the field of cardiology and detecting risk people for some anomalies, even though they are asymptomatic at the time of the examination [3].
- And many more.

So, in general, this practice allows the improvement of medical techniques while reducing costs. However, automatic ECG's analysis techniques have since a long time been based on static rules making them very inefficient when the recordings are not made in perfect conditions and totally inefficient when patterns to identified are not easily describable. With the advent of Deep learning, artificial intelligence has considerably improved the analysis of ECG recordings with the possibility of recognising much more complex patterns. However, at present, commonly used algorithms are still very sensitive to noise and poor quality signals. Thus, when analysing long time recording as Holter's data, cardiologists are forced to systematically mandate a technician to correct automatically generated annotations before analysing the results themselves.

The goal of this project is to apply state of the art deep learning algorithms to the annotation of electrocardiograms in order to allow a more robust and sensitive annotation, less affected by poor quality signals

II. RELATED WORK

Much work has already been done in the field of ECG annotation using deep learning. Awani et al. [2] have reach arrhythmia detection performances who surpass some reference specialists by using convolutional neural networks (CNN).

In the context of this sequential data analysis problem, recurrent neural networks have naturally also been

used, for example long-short term memory cells (LSTM) by Saeed et al. [4] who reach good performances with a model that has the particularity of being very light and suitable for processing data acquired in real time on portable devices with low computing power.

Since 2017 and Vaswani et al. [5], A considerable improvement in natural language processing performance has been achieved through the development and use of a deep learning architecture called Transformers. This architecture corrects various weaknesses of RNNs such as the parallelization of sequence reading operations, the absence of the use of a limited state of knowledge causing a very sharp drop in performance over long sequences, and also allows an excellent analysis of the general context of sequences through the use of powerful attention mechanisms. This approach has already been applied in the context of heartbeat classification by Yan et al. [6].

III. METHODS

In this project, we will implement and use the transformer architecture to successively solve the following two problems:

- Segmentation of ECG recordings beat by beat: To do this, we will train a transformer on the task of ECG's R-waves detection with the best possible accuracy.
- Heartbeat annotation: From the segmented recordings, this second transformer will annotate each heartbeat to determine if it is normal or if it is not, which family of abnormality it belongs to.

A. Basis data format

There are an infinite number of possible formats for storing and analysing ECG recordings. In this project we will be confronted with two main formats:

- The MIT format: This is the format use by the reference open-source dataset. Samples are float values who respect the millivolt (mV) scale, encoded with 11 bits values and recorded with a rate of 360 samples per seconds.
- The second is the format that are provided by Bittium recorders who are use by the University hospital of Liège. This samples are encoded by integers on the micro-volts (µV) scale and a sample rate of 250 Hz.

Given the more up-to-date aspect of the second format, as well as the fact that we have a much larger quantity of recordings from Bittium recorders, it is this second one that we have chosen as input for our algorithms.

From a practical point of view, we have chosen to work in a general way using the European Data Format (EDF) to store our records on disk. This format allows us easily to deal with multi derivations algorithm and corresponding annotations intuitively by the existence of several open-source libraries. In addition, the EDF-Browser software, also open-source, provides a convenient graphical interface for the analysis of our files, in order to best prepare our learning sets and analyse the results provided by our algorithms.

B. General Transformer's architecture

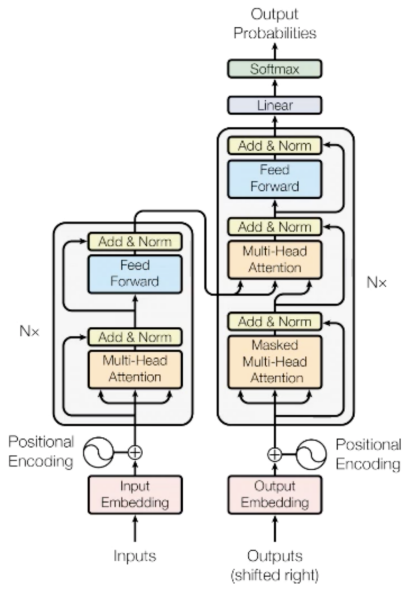

FIG. 3: Transformer's global architecture [5]

The original shape of the transformer's implementation, as describe by Vaswani et al. [5] and illustrated in the figure 3. This Transformer is composed of two main parts: The encoder which takes as input a sequence of embedded representation of our inputs, and the decoder which takes as input the representation of our inputs provided by the encoder, and an output-like value, which classically can be the embedding of the last generated work if we are in the case of a sequence to sequence NLP problem.

During this project, and for the two tasks we will be doing, only the encoding part will be useful. This part, whose implementation will be discussed in more details in the next section, consists of the repetition of a number of blocks composed of two main layers. The first layer consists of a self attention layer which, by means of simple matrix operations, will allow the model to create links between the different elements present in our sequence. The second layer consists of a simple feed-forward layer allowing the model to memorise the knowledge necessary for an adequate representation of the data. After each of

these layers, a normalisation is carried out to allow the model to correctly generalise the acquired knowledge.

By using these succession of multi-head self attention and feed forward layers, our model is then able to generate a representation of the inputs allowing us to recognise with great abstraction the patterns necessary for our annotation task. Thus, the data generated by the encoder can be concatenated and reused by simple feed forward layers as a decoder.

C. General model Implementation

1. *Data pre-processing and input embedding*

First of all our raw data need to be pre-processed in order to remove as much imperfection as possible such that our model is able to focus on what a prediction really depends on. ECG signals can be subject to a great variety of artifacts. The literature propose a plethora of possible filters and method to purify our signals. We have decided to opt for a method simple to implement, proposed in [7], which provide good results. We have applied the following transformation:

1. A first median filter, with a sliding window taking into account the direct neighbors of a time value.

2. A second median filter, used on the resulting signal. This one is extended to a window considering seven values.

3. A 12 order low-pass filter with a 35Hz cut-off frequency, implemented using the scipy library.

Where we define a median filter by:

If n is odd,

$$y(i) = median\left(x(i - \frac{(n-1)}{2}) : x(i + \frac{(n-1)}{2})\right)$$

If n is even,

$$y(i) = median\left(x(i - \frac{n}{2}), ..., x(i + (\frac{n}{2}) - 1)\right)$$

At that point we can hope that the signal has been cleaned and that the important features are more likely to be captured during the training phase. In addition, if necessary, the data is re-scaled, as described in the previous section, and re-sampled from 360 Hz to match 250Hz.

Classicaly, in NLP, transformers are taking as input a sequence of vectorial representation of each words of the sequence, called embedding. In our case, we have decided to add at the start of our model a one dimensional convolutional. This layer will take as input our ECG signal of n samples, use a kernel of size 50, with an appropriate padding in order to generate a vectorial representation of $d_{model} = 20$ values for each sample. So, our embedded version of the signal is now a $n \times d_{model}$ 2d tensor.

Unlike RNNs, self attention mechanism are permutation equivalent, so transformers basically do not take into account the position of the various inputs within the sequence read. In this project, we have made the same choice as Yan et al. [6] did, by using a sinusoidal positional encoding. This strategy exploits the periodicity of our data in order to handle input sequences which are longer than sequences used during the training process.

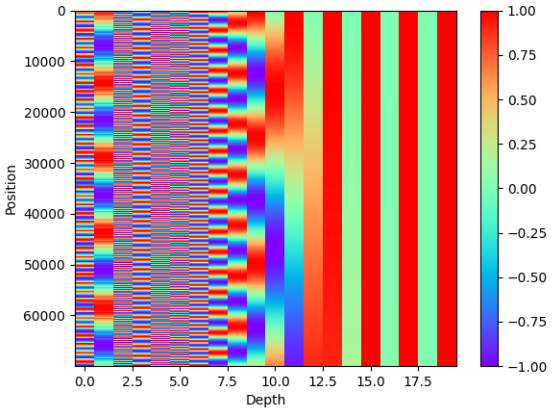

FIG. 4: Positional encoding values for $d_{model} = 20$ on sequences of max 70.000 samples

Therefore, our implementation of the positional encoding works by adding an additive value to the input matrix using the following formulas:

$$PE_{t,2i} = sin(\frac{t}{10000^{\frac{2i}{d_{model}}}})$$

$$PE_{t,2i+1} = cos(\frac{t}{10000^{\frac{2i}{d_{model}}}})$$

Values for our $d_{model} = 20$ and a max sequence length of 70.000 samples are presented in the figure 4.

2. *Multi-head self attention*

The core of our model is based on a Scaled dot-product attention mechanism. During this step, which is illustrated in the figure 5, input data are represented in three matrices: Value (V), Key (K) and Query (Q). Because we are in a self attention mechanism, both are built from the same input matrix of size $n \times d_{model}$ which are passed trough a specific simple feed forward layer for each matrix. After that, the scaled dot-product mechanism consists of the simple following matrix multiplication an re-

scaling operations:

$$attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

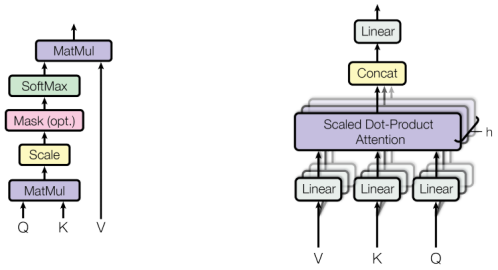

FIG. 5: Scaled dot product (left) and Multi-head attention (right), Vaswani et al. [5]

In our implementation, which follows the transformer architecture [5], we are using a variant of this attention mechanism called Multi-heads attention and which is shown in the right part of the figure 5. When the three matrices are built, our $SelfAttention$ class will reshape them according to our $h = 5$ number of heats. Outputs of this multi-heads attention module are concatenated and processed by a simple feed forward layer.

By the same way that in the figure 3 we have implemented a skip connection by adding the value of the input Querry matrix before the normalization step. This permit that the gradient pass in every conditions.

3. *Final encoder structure*

To summarize, both models that we are using use the same encoder architecture: A succession of four encoder block which:

- Take as input sequences with an embedding size of 20
- A forward expansion of 6 to give hidden layers of 120 neurons for the feed forward steps. These layers are using ReLU activations.
- A dropout value of 0.1 which is applied during both learning processes
- A layer normalization after each attention block and each feed forward layers
- Two skip connections to bypass the multi-head attention and the feed forward steps for the gradient signal.

4. *Decoder Structure*

For the two models we have developed, the decoder architecture follows more or less the same rules and is based on four fully connected layers. The three first are using the ReLU activation function. Hidden layers have a size of $d_{model} \times forward\_expansion$, so 120 neurons.

For the Beat Classifier model, which has to classify the beat, this decoder layer takes as input the concatenation of the whole matrix which outputs from the encoder. So this strategy mandates us to feed to this model only sequences of beat that are sharing the same length of 350 samples. for outputs, the model returns a sigmoid activation and a softmax on 9 output values which represent the probability that the model associates to each beat classes.

For the segmenter model, the strategy changes by the fact that the decoder layers are taking as input separately each column's vector output from the encoder. By this way, the model returns an only value with sigmoid activation for each sample fed into the model. So, this algorithm can take inputs as long as desired, according to the memory capacity.

5. *Final global structure*

The final global structure, who represent the successions of our two deep learning model in order to annotate a raw ECG is synthesized in the figure 10.

More information about our implementation, as well as our source code and the weights associated with our models are available at the following Git-hub repository: `https://github.com/francoislievens/INFO8010-Deep_Learning_Project`.

D. The Segmentation model

The classification of heartbeats is only possible after identifying the heartbeats and segmenting the signal beat by beat. The majority of the publications we were able to review used simple algorithms consisting of detecting the peaks of the R-waves. This simple approach can be effective in the context of very good quality ECGs, since this R-wave is always the highest. However, in real life recordings such as Holters, this system quickly gives very poor results due to noise and interference in one or more leads.

We therefore decided to split this project into two phases, the first of which consists of training a transformer to robustly detect R-waves, thus allowing good quality segmentation of the ECGs, which can then be annotated by our second algorithm whose aim will be to classify the beats.

To solve this task, we are using our basis implementation of the transformer's encoder previously exposed, who take as input sequences of ECG recordings, on which we are plugging as a decoder four simple feed-forward fully connected layers. So, our encoder provides us a representation of our input data which has a shape of $n \times d_{model}$, and so we obtain the same number of vectorial

representations as inputs. Each of these representations passes through our feed forward model where the three first layers are using the ReLU activation and a forward expansion of 2. The last layer returns a unique value with a sigmoid activation in order to reconstitute a new signal representing the meaning of the model about the presence of the R wave in each place of the sequence.

From this new signal, we can use a very simple peaks annotator like the one implemented in the sklearn library to detect evidence of peak certainty.

To train this model, we used nearly 200 3-lead Holter recordings recorded with Bittium sensors by the cardiology department of the University Hospital of Liege. These recordings last between 24 and 72 hours and are performed on patients with suspected heart rhythm abnormalities. As the patients go on with their lives in a normal way during the recording, we can assume that they allow us to cover a wide range of heart rates and types of abnormalities that can be encountered. These recordings also have the peculiarity of having a lot of recording anomalies such as noise and bias due to sensor movement or interference. Thus, large areas of recordings are completely uninterpretable, and regularly only one of the three leads has an interpretable signal.

Given the prohibitive cost of manual annotation by data, we opted for a sub-optimal strategy for training our algorithms: indeed, our learning set is composed of data automatically annotated by the software provided by the firm Bittium, but corrected by a specialised employee which erased the majority of the uninterpretable ranges.

Our *BittiumPreparator* class will so open raw EDF files and associated automatically annotated/manually corrected annotation, perform our de-noising preprocessing and conversion, and segment recordings into 30 minutes one lead annotated records. In this dataset, noisy parts are preserved with no annotations in order to train our algorithm to make the difference between noisy channel and real R-waves. The different leads of each recording are split into different annotated records sharing the same annotations, in order to increase our data. But since some leads are currently too noisy, we are deleting the ones having a too high noisy index provided by the Bittium software.

For the training of this segmentation's algorithm, we are generating a new annotation form: initially, the R-wave position is designed by a time value in millisecond. During our dataset building, we are converting this in a signal of the same shape of the input signal. This signal is initialized at a constant value of zero, before the adding of normal distributions, $\mathcal{N}(i, \sqrt{8})$ where i is the index provided by the R-wave annotation.

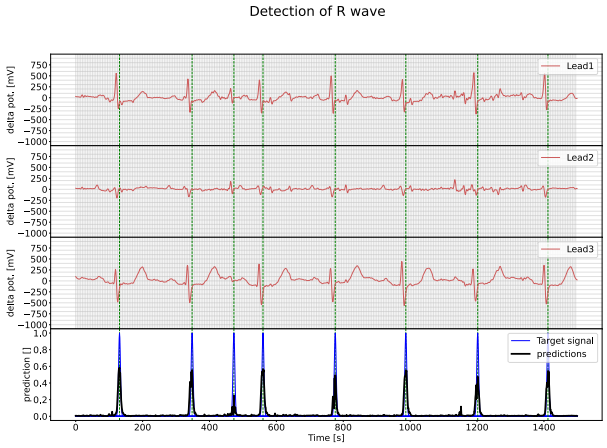

FIG. 6: R-wave annotations: input and target data

In the figure 6 we can observe targets and predictions of the R-confidence signal of our model for a really bad quality ECG signal with 3 leads.

A major improvement of our algorithm next to classical peaks finding is that we are able to deal with multiple leads signal, even if some of them are only noise. So, our model will generate a R-confidence signal for each input signal and then construct the max prediction signal on which the peaks detection is performed. So, we can see in the figure that despite some signals give less information on the position of the wave, the first lead gives an almost perfect prediction. We can also see that although the discrimination of the model seems very good, small peaks that are not corresponding to real R-wave are present. Our model has two hyper-parameters to fight against false positives: The first is a minimum distance between two peaks, and the second is a threshold which corresponds to the minimum level of certainty to be reached before annotating a certainty peak as an R-wave.

E. Classification model

1. *Segmentation strategy*

The majority of publications working on cardiac arrhythmia's detection cut their signal to obtain R-wave to R-wave segments. Indeed, R waves are strongly marked, whereas the actual separation between two cardiac revolutions is much more blurred to determine. However, we chose not to follow this strategy because in this way the segments cover the second half of one cardiac revolution, and the first half of another. It seems to us much more logical to annotate pathological beats, which can be isolated, by analysing the whole pathological heartbeat on the same signal because patterns on the first part of the beat can be linked with patterns on the next part.

Thus, to overcome the difficulty of detecting separations between beats, we decided to set a length of 350 samples per beat, which is longer than any full heartbeat of very low frequency, and to use as input to our annotation algorithm each beat of this length, centred on the corresponding R-wave annotation. This gives us the facility to work with all beats of the same size, but also gives us a little more information about the overall context of the beat.

2. *Dataset*

Since we unfortunately have neither the time nor the means to generate a better manually annotated dataset, we have decided to limit ourselves to the MIT-BIH [8] [9]. This is an old database of 1980's which contains two-channel ambulatory ECG recording by the Beth Israel Deacones Medical Center and the Massachusetts Institute of Technology. This popular data bank contains a total of 24h of two leads ECG's sequences recorded on 47 subjects for at the end of the day 110.000 cardiac beats manually annotated.

After the pre-processing of our data, performed by our class $MIT_Dataset$ and which consists in a rescaling process in the uV scale and a re-sampling to 250Hz, beats corresponding to classes with less than 1000 occurrences in the dataset are deleted to keep only classes presented in the table I.

Sym.	Arrhythmia type	occ.
· or N	Normal beat	150032
L	Left bundle branch block beat	16144
R	Right bundle branch block beat	14512
A	Atrial premature beat	5088
V	Premature ventricular contraction	14260
F	Fusion of ventricular and normal beat	1606
/	Paced beat	14048
!	Ventricular flutter wave	944
f	Fusion of paced and normal beat	1964

TABLE I: MIT-BIH / WFDB annotations symbols meaning

While those training beats are segmented using the strategy developed in the previous section, beats are randomly shuffled and the dataset is split in a training set containing 80% of the dataset and a testing containing the remaining 20%.

3. *Model's structure*

The general structure of the encoder is exactly the same as for the segmentation model but, this time, the transformer can only take as input sequences of the fixed size of 350 samples, centered in the R-wave of the beat. By this way, outputs of the encoder already have the same shape of 20×350 values that can be concatenated in a 1D vector to pass through four feed-forward layers. These layers keep the same shape as in the network that we are using for the segmentation, but end by a softmax on a vector of size 8, in order to return a probability value for each class of the dataset.

IV. ECG'S SEGMENTATION: TRAINING AND RESULTS

A. Training Process

As already mentioned, the dataset we use is composed of 200 long-duration ECG records, pre-processed by our BittiumPreparator class. This alogorithm provides us a folder containing a very large number of serialized tensors which memorize a single 30-minute lead recording with the associated target R-confident signal.

Of these files, 20% are kept as test sets and the remaining 80% are used as training sets. To manage the reading of this dataset during training, we have implemented the $BittiumTrainSetBuilder$ class. At each iteration, this data handler loads 100 randomly chosen files in memory to be browsed by the learning algorithm. During the learning phase, the heartbeats from all loaded sequences are mixed together.

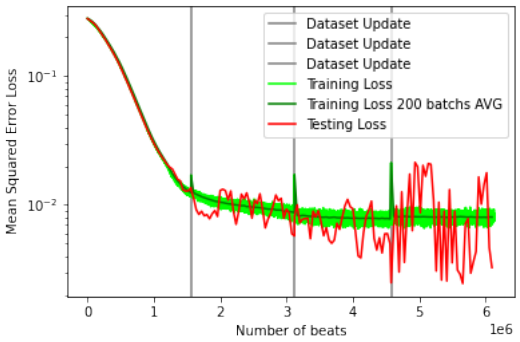

FIG. 7: R-wave Annoter model training (logarithmic scale)

The tracking of train and test loss of our model during the training process are presented in the figure **??**. During this training, we are using:

- The Adam optimizer
- A learning rate of $5 * 10^{-6}$
- The mean squared error computed between the predicted signal and the target signal.

A remarkable feature is the incredible learning speed of our model. The multi-head self attention mechanism allows to quickly capture the links between the inputs and excellent performances are very quickly reached with

few data browsed. Thus, only a few minutes are needed to achieve acceptable predictions. However, this model takes up a lot of memory space and we were forced to limit the size of our batches to 200 record segments, each containing 350 samples. This high learning capacity coupled with the small size of the batches forces us to use the very low learning rate of 10^{-6} in order to allow our model to reach a loss of about 0.01.

B. Performances Analysis

In order to evaluate performances of our model, we have chosen the MIT-BIH dataset, which was not used in the learning set.

We then implemented a method, which uses our model to predict the R-waves of the test recordings by varying our confidence threshold (i.e. the minimum confidence of the model to determine that a peak confidence corresponds to an R-wave) from 0 to 0.5 in steps of 0.02. At each iteration, the true positive, false positive and false negative rates are computed.

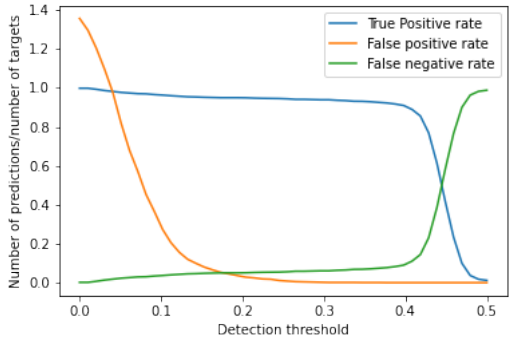

FIG. 8: Beat Classifier: Training

Results of this experiment are shown in the figure 8. This figure highlights the fact that when we are using our algorithm to predict R-waves, increasing the value of our confidence threshold from 0 to 0.4 doesn't affect a lot the true positive rate but significantly improves the number of false positive waves detected. So, depending on the use of the algorithm, it can be more interesting to obtain as few undetected beats as possible, even if the number of false detection is high, or vice versa. But in our case, we think that a threshold value around 0.3 gives a really good deal.

V. BEATS CLASSIFIER: TRAINING AND RESULTS

A. Training Process

The training process is organized as follow:

- Our model predicts probabilities for each of the 9 classes that we kept.
- We are using the CrossEntropy loss function. Due to the fact that the distribution of beats classes are unbalanced, we are weighting the loss computed on our batches by using weights associated to each class. These weights are the inverse of the proportion of this class in the dataset. So, if our model predicts a vector of I values p, for a target class $c \in [0; I]$, and that the number of occurrences of the class c in the dataset is n_c, the loss is computed as follow:

$$loss(p, c) = -\log\left(\frac{\exp p_c}{\sum_{i=0}^{I} \exp p_i}\right) * \frac{\sum_{i=0}^{I} n_i}{n_c}$$

- Once again, we are using the Adam optimizer.
- A learning rate of 10^{-5}

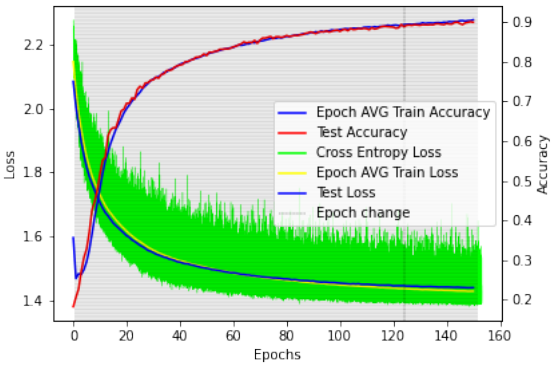

FIG. 9: Beat Classifier: Training

The performances tracking of our algorithm during the training is shown in the figure 9. In this figure, losses are weighted crossentropy loss, but accuracy's are computed by averaging the binary vector containing good predictions of the model and so are computed on the true distribution of the dataset.

So, after 150 epochs on the MIT-BIH dataset, our final model reaches an accuracy of 0.90 as well on the test set and the training set, and a crossentropy loss around 1.44 on the test set and 1.43 on the training set. We have decide to stop the training by lack of time, but the model was still not overfitting after these 150 epochs.

B. Performances Analysis

The ROC curves associated to each classes are show in the following appendix:

- Ventricular flutter wave 12
- Atrial premature beat 12

- Fusion of paced and normal beat 13
- Left bundle branch block beat 14
- Normal beat 15
- Right bundle branch block beat 16
- Paced beat 17
- Premature ventricular 18

We can see that the capacity of the model to recognise some patterns depends as much on the pattern itself than on the number of occurrences of that pattern in the training set. Indeed, we can notice that the Ventricular flutter wave class, although its minority (944 occurrences), is very well detected, whereas the "Fusion of paced and normal beat" class is less well detected with more than twice the number of occurrences (1964).

VI. FINAL RESULTS

A final overview of the combined work of our two algorithms to segment and annotate the heartbeats of a two-lead ECG extract from the MIT-BIH database from the raw signal is presented in figures 19, 20, 21 and 22. In these figures are plotted the two derivations of the ECG sample, the "confidence R-wave signal" generated by our segmentation model, vertical lines represent final R-wave annotation, and the final labels predicted by our classifier model, as well as the original target annotation.

VII. CONCLUSION AND DISCUSSION

During this project, we managed to develop a complete algorithm, i.e. to annotate a file that has not been previously segmented. Both problems were successfully solved based on the Transformer architecture which we customised to our needs. Although the two tasks are completely different, we managed to solve them with very little change to the architecture of our basic encoder. The more classical classification problem was easily solved by using weighted cross-entropy loss.

The segmentation problem, on the other hand, required us to be more imaginative and we solved it by creating a "target-signal" by centering a normal distribution on the indexes corresponding to the annotations of the R waves. Using this technique, our algorithm predicts QRS complexes much more reliably than most simple algorithms that directly detect the initial signal peaks and become totally ineffective when using wearable devices who records really noisy signals.

We were impressed by the speed of learning and the ability of the multi-head attention mechanisms to detect patterns on ECGs. Moreover, the performance of our algorithm seems to be highly generalizable when used to annotate signals from devices with very different qualities and amplitudes or number of leads.

The performances obtained on the MIT-BIH dataset are quite close from state of the art performance found in comparable literature, but not quite as good with an accuracy of 0.9. Our weakly poorer performances can be explained by the fact that we ended the learning phase of our classifier when it seemed to be able to improve its capabilities slightly. Moreover, the MIT-BIH dataset is relatively small, and we chose to keep 20% of the data as a test set. We could have opted for a different validation system to take advantage of a larger part of our data during the training phase. It is also important to note that most publications using MIT-BIH as a reference are limited to the 4 classes with the most occurrences in the dataset whereas we have kept 9.

Our Achilles heel is undoubtedly the training data. For the segmenter, we used data annotated by a commercial model with a human global correction consisting of deleting annotations of contentious record portions. However, this strategy seems to have been globally effective, probably due to the very large amount of data we had at our disposal. However, regarding the heartbeat classification problem, extracting a reliable dataset from the data we had at our disposal would have required considerable time and work. We therefore chose the strategy of limiting ourselves to the MIT-BIH database.

We had initially decided to apply unsupervised learning techniques on the large amount of data obtained from the University Hospital, in order to perform transfer learning and fine-tuning on the MIT-BIH data in order to make the best use of the latter and to generalise them in the best way. However, we did not have the time to finalise this approach within the time limits of the project, but we still have the opinion that this approach can provide good results.

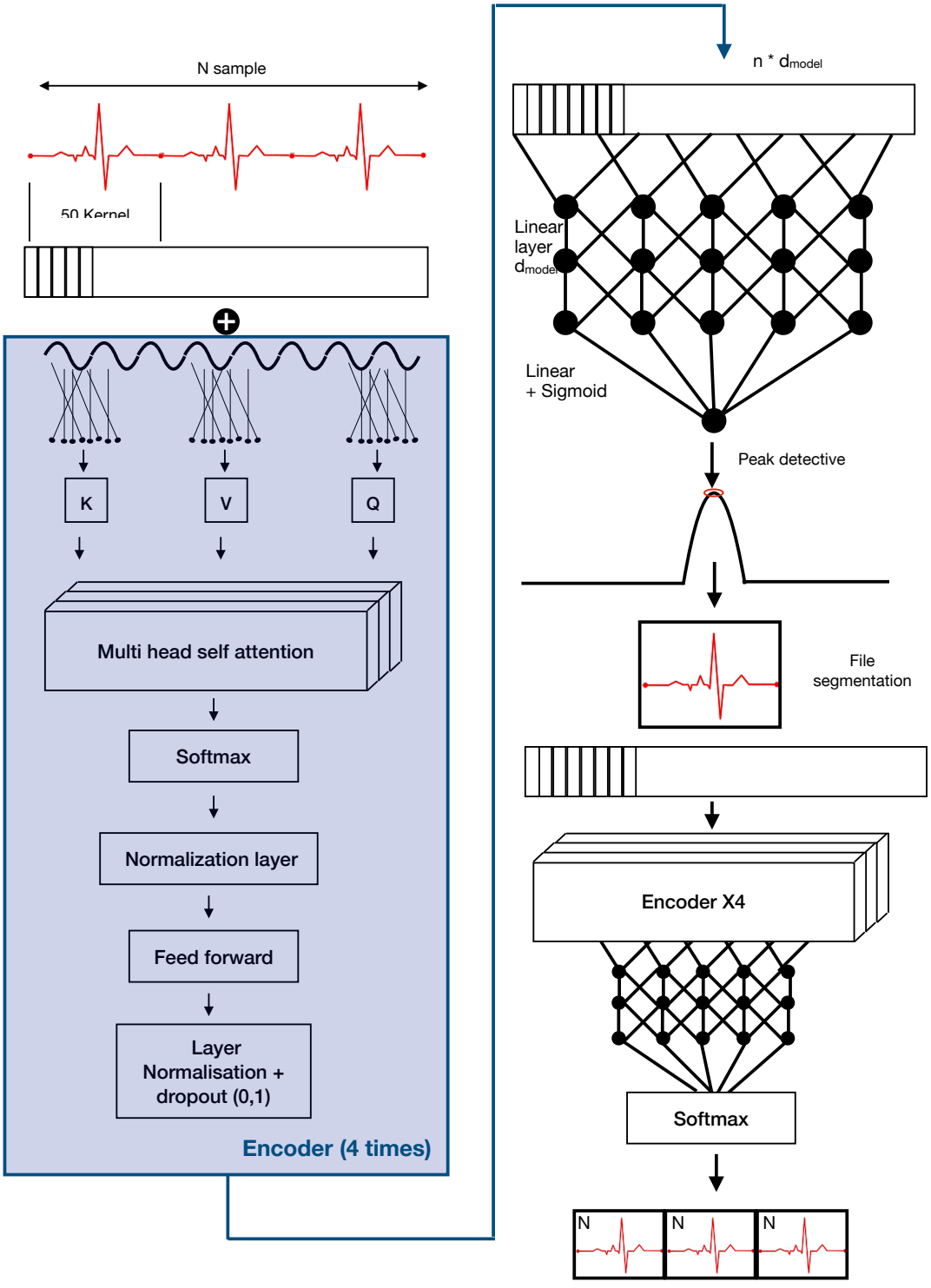

FIG. 10: General Algorithm shape

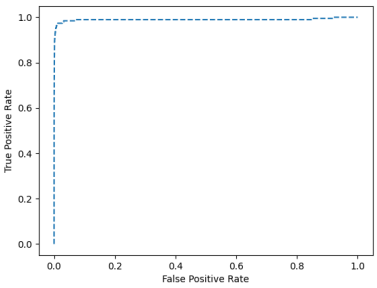

FIG. 11: ROC curve Ventricular flutter wave

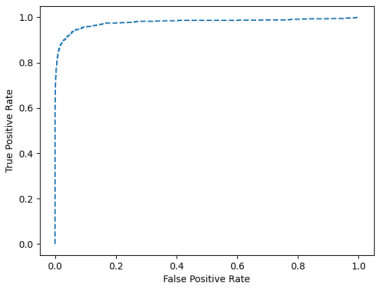

FIG. 12: ROC curve Atrial premature beat

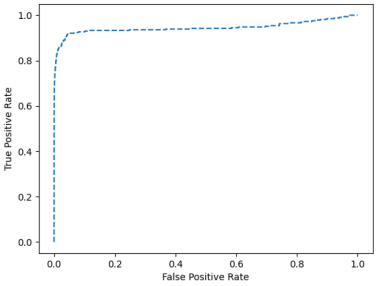

FIG. 13: Roc curve Fusion of paced and normal beat

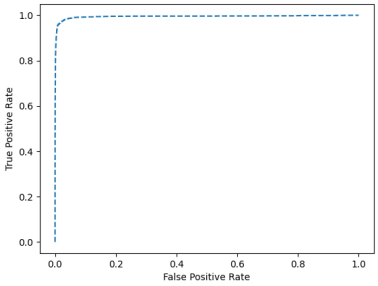

FIG. 14: ROC curve Left bundle branch block beat

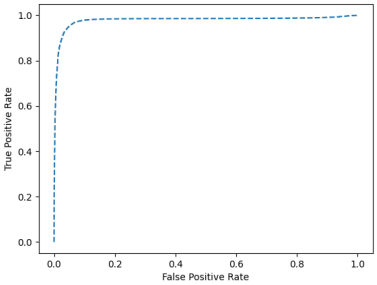

FIG. 15: Roc curve Normal beat

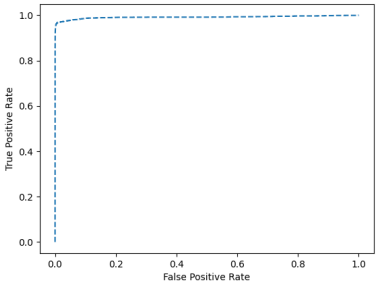

FIG. 16: Roc curve Right bundle branch block beat

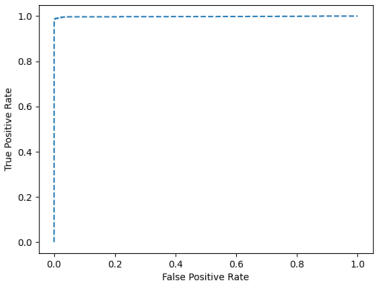

FIG. 17: ROC curve Paced beat

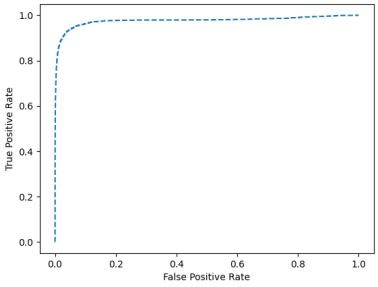

FIG. 18: ROC curve Premature ventricular contraction

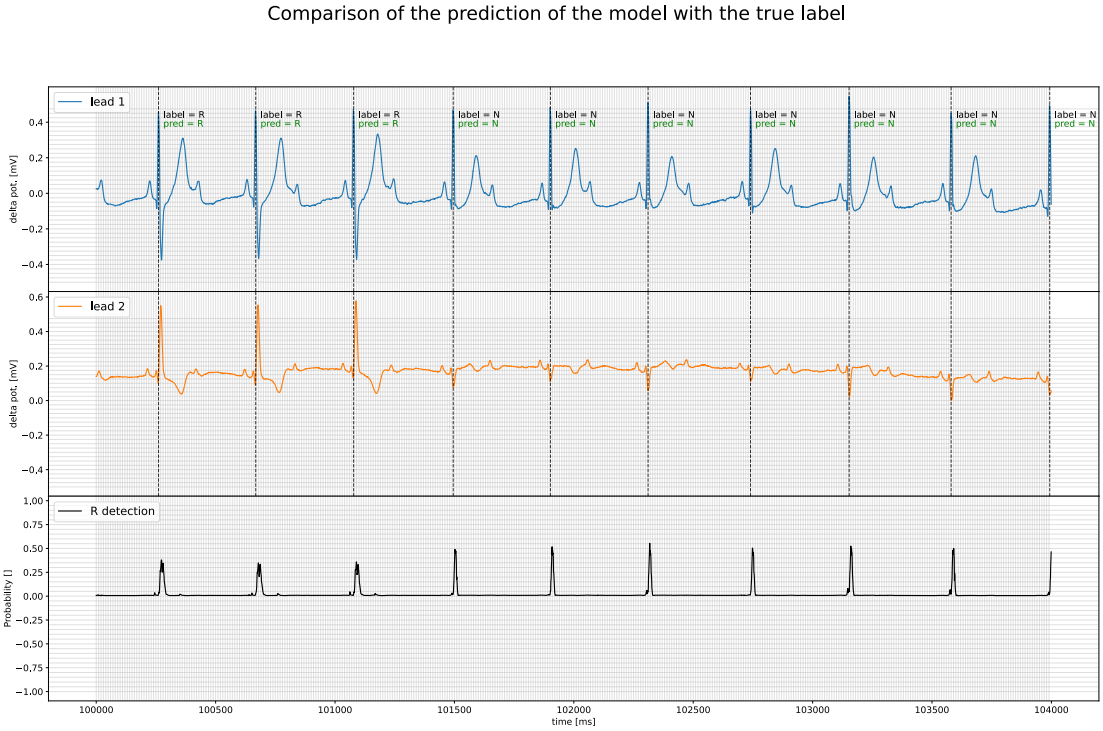

FIG. 19: Final demonstration of the algorithm on a sample from the MIT-BIH dataset.

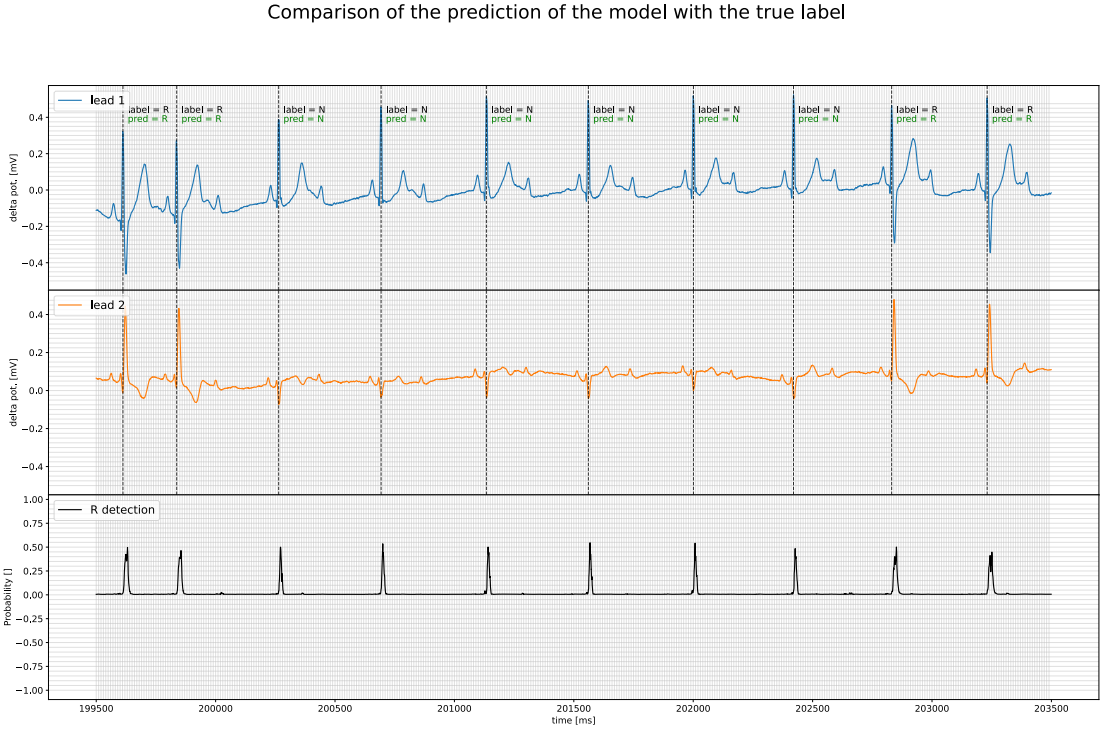

FIG. 20: Final demonstration of the algorithm on a sample from the MIT-BIH dataset.

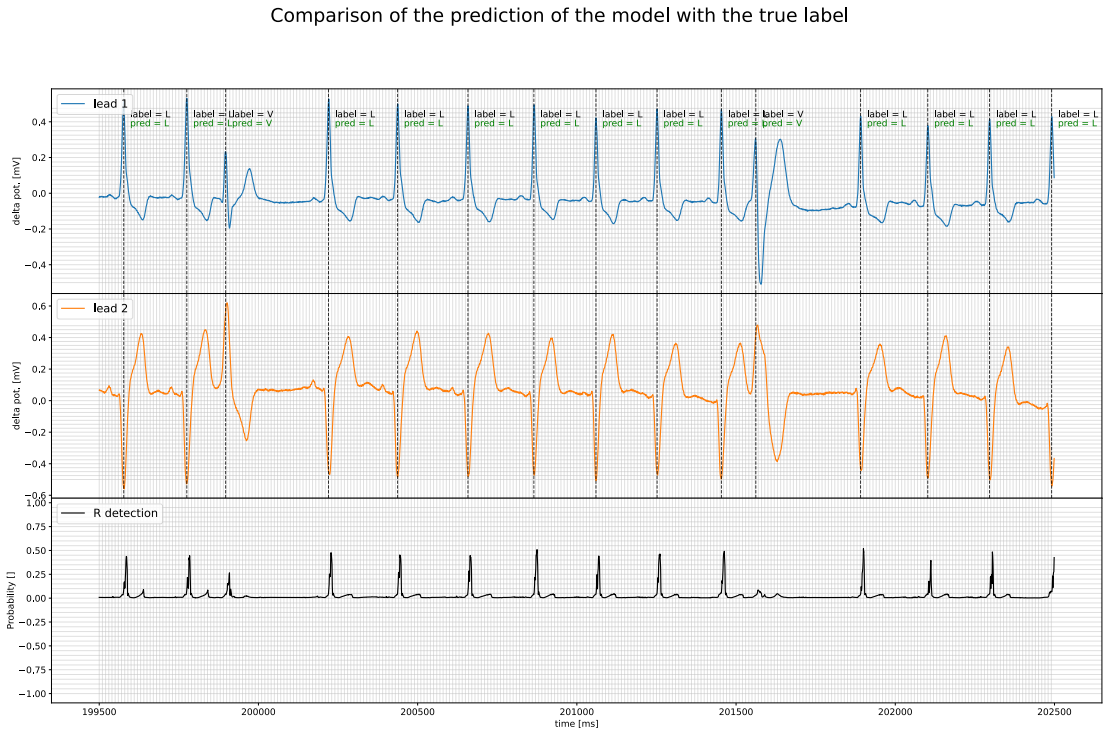

FIG. 21: Final demonstration of the algorithm on a sample from the MIT-BIH dataset.

FIG. 22: Final demonstration of the algorithm on a sample from the MIT-BIH dataset.

[1] J. Espinosa-Oviedo F. J. Martínez-Tabares and G. Castellanos-Dominguez. Improvement of ecg signal quality measurement using correlation and diversity-based approaches. pages 4295–4298.

[2] Pranav Rajpurkar Awni Y. Hannun. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. 25:65–69.

[3] Alvaro E. Ulloa-Cerna Arun Nemani Tanner Carbonati Linyuan Jing David P. vanMaanen Dustin N. Hartzel Jeffery A. Ruhl Braxton F. Lagerman Daniel B. Rocha Nathan J. Stoudt Gargi Schneider Kipp W. Johnson Noah Zimmerman Joseph B. Leader H. Lester Kirchner Christoph J. Griessenauer Ashraf Hafez Christopher W. Good Brandon K. Fornwalt Christopher M. Haggerty Sushravya Raghunath, John M. Pfeifer. Deep neural networks can predict new-onset atrial fibrillation from the 12-lead ecg and help identify those at risk of atrial fibrillation–related stroke. 143:1287–1298.

[4] M. Oveisi S. Saadatnejad and M. Hashemi. Lstm-based ecg classification for continuous monitoring on personal wearable devices. 24:515–523.

[5] Niki Parmar Jakob Uszkoreit-Llion Jones Aidan N. Gomez Łukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. 30.

[6] J. Espinosa-Oviedo F. J. Martínez-Tabares and G. Castellanos-Dominguez. Fusing transformer model with temporal features for ecg heartbeat classification. pages 898–905.

[7] G. De Pietro G. Sannino. A deep learning approach for ecg-based heartbeat classification for arrhythmia detection. 86.

[8] Glass L.-Hausdorff J.-Ivanov P. Goldberger A., Amaral L. The impact of the mit-bih arhythmia database. 20:45–50.

[9] Moody G.B Mark R.G. Evaluation of automated arrhythmia monitors using an annotated ecg database. 37:205–210.