Exploration of supervised machine learning models to detect epileptic crises from Heart Rate Variability features

João Saraiva
IST-86449

Débora Albuquerque
IST-75864

Course / School
Machine Learning in Bioengineering
Department of Bioengineering
Instituto Superior Técnico

***Abstract*—There has been an interest to detect, predict and monitor epileptic seizures in patients with refractory epilepsy. Epileptic crises have been associated with modifications in the autonomic nervous system (ANS), which usually precede the seizure onset for several minutes and also manifest during the crisis itself. This insight leads to the suggestion of using machine learning models to try to detect seizures solely with heart rate variability (HRV) features. In this work we explore the use of supervised learning models to detect epileptic seizures, namely Support Vector Machines (SVMs) and Temporal Convolutional Neural Networks (1D-CNNs). Our SVM classifiers achieved an average detection accuracy, F1-score and false positive (FP) rate of 94.9%, 75.6% and 2.6%, respectively. Our CNN classifiers can accurately detect (almost predict) seizures 15 minutes before onset, when designed with the right architecture, and present a false positive rate of 25% minimum. These results are promising, but more work should be conducted to polish the CNN architectures in order to see if they can actually help with this problem or not.**

***Index Terms*—SVM, CNN, HRV, ECG, seizure detection**

I. Introduction

Epilepsy is a neurological disorder characterized by an enduring predisposition to generate epileptic seizures and the associated cognitive, psychological and social consequences [1]. The crises, called epileptic seizures, afflict about 1% of the world's population. Approximately, anti-epileptic (AE) drugs are able to control seizures of 2/3 of all epilepsy patients. The other 1/3 of patients are called drug-resistant epilepsy patients and are said to have refractory epilepsy [2].

Generally speaking, an epileptic seizure is manifested by an abrupt change in behaviour, either semiologically or not, caused by abnormally excessive or synchronous neuronal activity in the brain. As a result, skeletal muscles of a subject having a seizure may contract involuntarily [3]. Common signals and symptoms might be loss of awareness, stiffening, jerking, a sensation that rises from the abdomen to the chest, and a smell of burnt rubber [1].

In pathophysiology, ictogenesis regards the transition between an interictal period and the ictal onset, progression and termination [4]. An EEG signal in a crisis segment may be divided into interictal, preictal, ictal and postictal periods. The ictal phase is the outbreak period of epilepsy. The phases immediately before and after the ictal phase are the preictal and postictal phases. Finally, the interictal phase is the period between seizures [5].

Studies in rat models have shown significantly different ictogenic networks for focal and generalized ictal onset seizures [6]–[8]. So, this became the first question to answer when classifying seizure types. In 2017, the international league against epilepsy (ILAE) reviewed the classification of seizure types [9], and in this classification, seizures are first discriminated by their onset type: focal, generalized or unknown:

- **Focal seizure**[1]**:** Originates in one or more localized parts of the brain.
- **Generalized seizure:** Originates from widespread regions on both hemispheres of the brain.

Besides the temporal periods aforementioned, seizures can be divided into phases regarding the symptoms manifestation on the patient during a crisis. The first stage is the aura phase, which temporally corresponds to the preictal period and can last from a few seconds to an hour in duration [10]. Then in the ictal period, there can be tonic or clonic phases, or both. A tonic seizure is accompanied by sustained muscle contraction. A clonic seizure is accompanied by rhythmic muscle jerks. A tonic–clonic seizure begins with a muscle contraction (tonic phase), usually causing the patient to fall, followed by rhythmic muscle jerks (clonic phase) [2]. And then comes the postictal phase after a few seconds or minutes. Some symptoms of each of these phases are enumerated below:

1) **Aura:** Fatigue, dizziness, numbness, confusion, hallucination, unusual sounds, distorted emotions, distorted taste, or tingling sensations.
2) **Tonic:** Back arched, stiff body, epileptic cry, incontinence.
3) **Clonic:** Jerking movements, blinking eyes, frothy saliva.
4) **Postictal:** Weak limbs, exhaustiveness, sleepy.

Not every crisis comprehends all these stages. More types of seizures include **hyperkinetic seizures**, which are accompanied by intense motor activity; **myoclonic seizures**, which are accompanied by rapid, involuntary muscle twitches; **reflex seizures**, which are elicited by a specific stimulus, such as flashing lights, hot water or reading; and **automatisms**, which are unconscious behaviours that can occur during some epileptic seizures, such as lip smacking and hand fidgeting, picking or rubbing [2].

Recently there has been a particular attention in studies focusing on the heart rate changes to detect, predict and monitor epileptic seizures. The hypothesis behind this is that seizures are associated with modifications in the autonomic nervous system (ANS), which controls the heart rate [11]. This has led to the development of algorithms for classifying and predicting epileptic seizures using the electrocardiogram (ECG), and more specifically, heart rate variability (HRV) features extracted from it. The features that can be extracted might be of the time domain, frequency domain or non-linear features. Next we introduce each of these that we used in our work.

A. Time Domain Analysis

In time domain analysis, we are interested in the beat-to-beat amount within an interval of time. The features of this type of analysis can be divided into two classes: (i) those derived from direct measurements of the NN intervals (NNI) or instantaneous heart rate, and (ii) those derived from the differences between NN intervals. The most common extracted features in the time domain analysis are the mean NNI, i.e. the mean value of the NN intervals; the root mean square of successive differences (RMSSD) between each heartbeat; the NN50, which is the number of interval differences of successive NNIs greater than 50 ms; and pNN50,

[1]Formerly (before 2017) called partial seizures.

which is the proportion derived by dividing NN50 by the total number of NNIs.

B. Frequency Domain Analysis

This type of analysis involves the study of the power spectral density (PSD) of HRV. The HRV spectrum can be divided into 3 main routine bands: the very low frequency (VLF) band from 0.003–0.04 Hz, the low frequency (LF) band from 0.04–0.15 Hz, and the high frequency (HF) band between 0.15 Hz and 0.4 Hz [12].

The physiological mechanisms responsible for activity within the VLF band remain uncertain, although it appears that the VLF rhythm is intrinsically generated by the heart, with its amplitude and frequency being modulated by efferent SNS activity, due to physical activity and stress responses [13].

And while LF serves as an indicator of the sympathetic modulation on the heart as well as parasympathetic influences related to the respiratory system, HF is related to parasympathetic activity mainly caused by respiratory sinus arrhythmia. The LF/HF ratio is also commonly used as a HRV feature, and gives us an indication of the balance between the activity of sympathetic and parasympathetic systems [12].

C. Non-linear Analysis

Non-linear phenomena are also involved in HRV. The most common non-linear indices include the sample entropy (SampEn) or the coefficient sample entropy (CoSEN), which reflect the degree of irregularity in a series of data, and the Katz fractal dimension that calculates the fractal dimension of a sample using the Euclidean distances between the successive points of the sample [12]. Another non-linear feature of HRV is the Poincaré plot that shows how well each NNI predicts the next. We can extract the SD1 and SD2 features of the Poincaré plot.

The features selected in the detection and prediction of seizures should be considered in regard to the type of seizure being analyzed, as each feature, or subgroup of features, is relevant for different seizure onset patterns. For instance, second generalized seizures present an increase in mean HR, LF/HF and SD2/SD1 ratio when compared to complex partial seizures [14]. Temporal lobe seizures are characterized by an increased heart rate (HR), when compared to extratemporal seizures [15].

...

II. Machine Learning Background

This Section describes the mathematical foundations behind the classifier techniques we used.

A. Support Vector Machines

A support vector machine (SVM) is a non-parametric mathematical framework excellent for when we do not have any specialized prior knowledge about a domain.

In order to separate example points into two classes, an SVM tries to create what is called a maximum margin separator [16]. This is a linear hyperplane with the largest possible distance between example points, which helps them to generalize well. Even if the n-dimension examples are not linearly separable in the n-dimensional space, SVMs can create this linear hyperplane by taking advantage of the so-called *kernel trick*. This is clearly an advantage of SVMs over linear or logistic regression models. Even in the 2-dimensional space, these models would mathematically classify all the examples correctly, minimizing the loss, but for new queries it could be the case that some inputs were incorrectly classified if the regression hyperplane was to be *very close* towards one of the classes of the examples. After all, regression models only try to minimize expected empirical loss on the training phase [16]. Since we do not know to which class do the future points we will want to classify belong to, instead, SVMs ensure that the separator is the farthest away possible from both example classes seen in the training phase, thus attempting to minimize expected generalization loss[2]. In other words, the separator is equidistant to the nearest point of each class – called the support vectors. Support vectors are the example points, one from each class, closest to the separator in the training phase [16]. They are called *support* vectors in the sense that they *hold up* the separating hyperplane. The straight distance of the separator to both support vectors is called the margin, hence the name maximum margin separator. Figure 1 (left panel) depicts these concepts schematically.

Through gradient descent there can be found a weight vector, $\mathbf{w}$, and an intercept, b, that for an arbitrary point $\mathbf{x}$, the separator would defined as the set of points

$$\{\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0\}$$

in the respective dimensional space [16]. Another way is to represent $\mathbf{w}$ in the dual representation, α, and through quadratic programming find the optimal α by solving Equation 1, where $\sum_j \alpha_j y_j = 0$.

$$\alpha = \underset{\alpha}{\operatorname{argmax}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j \cdot \mathbf{x}_k) \tag{1}$$

The weights α_j associated with each data point are [16]:

$$\alpha \begin{cases} > 0 & \text{for support vectors} \\ = 0 & \text{otherwise} \end{cases} \tag{2}$$

The relation between α and $\mathbf{w}$ is $\mathbf{w} = \sum_j \alpha_j \mathbf{x}_j$. Notice that the *input* of Equation 1 is the dot product of pairs of points j and k, that is, $\mathbf{x}_j \cdot \mathbf{x}_k$. Equation 3 describes the separator itself. For an input vector $\mathbf{x}$ [16]:

$$h(\mathbf{x}) = \text{sign} \left(\sum_j \alpha_j y_j (\mathbf{x} \cdot \mathbf{x}_k) - b \right) \tag{3}$$

So, the aforementioned *kernel trick* is the operation that allows an SVM to map each input vector $\mathbf{x}$ to a new vector $F(\mathbf{x})$ of a higher-dimensional space. In this higher-dimensional space, the data becomes *linearly separable*. But notice the separator is actually non-linear in the original space [16]. In general, if there are n data points, then they will always be separable in spaces of at least $n-1$ dimensions. So, doing the transformation, it will suffice to simply replace $\mathbf{x}_j \cdot \mathbf{x}_k$ in Equations 1 and 3 by $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$. However, the transformation F can become quite computational expensive, so the whole process can be more efficient if we replace $\mathbf{x}_j \cdot \mathbf{x}_k$ by what is called a Kernel function, $K(\mathbf{x}_j, \mathbf{x}_k)$. For example, in a 3-dimensional space if F is defined as [16]:

$$F = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \end{pmatrix}$$

we can take advantage of square root properties to get an equivalence to $F(\mathbf{x}_j) \cdot F(\mathbf{x}_k)$. In this example, we simply compute the dot product of the two input vectors to the power of two, which is what is called the kernel function [16]:

$$F(\mathbf{x}_j) \cdot F(\mathbf{x}_k) = (\mathbf{x}_j \cdot \mathbf{x}_k)^2 = K(\mathbf{x}_j, \mathbf{x}_k) \tag{4}$$

[2]This comes from some arguments in computational learning theory, stating that some examples are more important than others, and that paying attention to them can lead to better generalization [16].

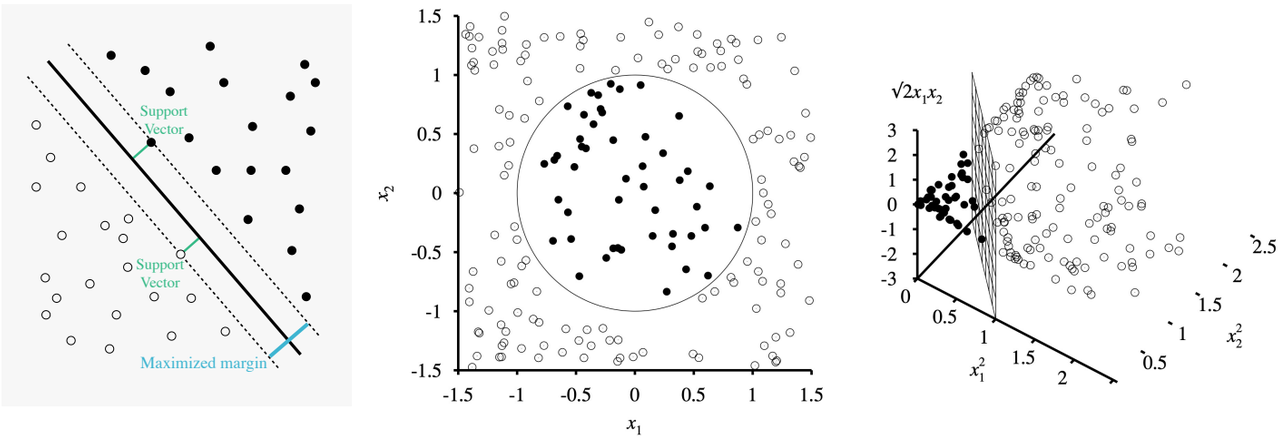

Fig. 1. Support Vector Machines learning theory. Left panel: Example of binary classification of examples into black dots or white dots. Support vectors (in green) are the points from each class closest to the separator. The margin (in blue) should be maximized as much as possible. Middle and right panels: Example of non-linear 2-dimensional classification. On the middle it is the 2-dimensional original representation, and on the right it is the 3-dimensional representation after applying the Kernel transformation. Notice how in the 3D space the two classes become linearly separable.

This *trick* will avoid applying F to each vector $\mathbf{x}$, by instead using the Kernel function $K(\mathbf{x}_j, \mathbf{x}_k)$, which allows us to create optimal linear separators efficient in feature spaces with billions of dimensions [16]. Figure 1 illustrates an example like this.

B. Convolutional Neural Networks

In general, artificial neural networks (ANN) are composed of **units** (or perceptrons) connected by **links** [16]. Between a pair of units i and j, a link between them can propagate an **activation**, a_i. Each of these links has a weight, w_{ij}, that indicates the strength and direction of the connection. In order to propagate the contribution of each unit i that is linked to a node j, the unit j computes the in-wards weighted sum of inputs [16]:

$$in_j = \sum_{i=0}^{n} w_{i,j} \cdot a_i \tag{5}$$

Then, the final activation of unit j, that determine if it fires or not, is given by applying the result of Equation 5, in_j, to the unit activation function, g. Equation 6 describes this activation [16]. The activation function, g, can be just a hard-coded threshold, where it is said to be activated when in_j surpasses that threshold, or a logistic function (sigmoid), hyperbolic tangent, or other modern functions, such as the Leaky Rectified Linear Unit (ReLu) function.

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} \cdot a_i\right) \tag{6}$$

Two layers of perceptrons can be stacked together, where the first holds arbitrary units i and the second holds arbitrary units j. In practice, multiple layers can be stacked together and we would called that type of networks as multilayer perceptrons (MLP) [16]. The input and output layers are the only *visible* layers, whereas the layers in between are said to be *hidden*. This organization in layers means that each unit receives input only from units in the immediately preceding layer (in feed-forward mode). And a MLP is said to be fully connected if each perceptron in one layer is connected to all perceptrons in the next layer. In this work we will use fully connected layers, but also convolutional layers and pooling layers, which can be found in typical convolutional neural networks (CNNs).

CNNs can be seen as regularized versions of MLPs. They were initially thought to solve the problem of identifying what objects are in 2 dimensional images. In the 1960s, two scientists, Hubel and Wiesel, showed that the visual cortex of monkeys have neurons that individually respond to small regions of the visual field. And neighboring neurons have similar and overlapping receptive fields. Later in the 1980s, Kunihiko Fukushima stated that what these visual cortex neurons are doing is basically a convolutional operation with their receptive inputs. So, that is the main idea behind using the convolution operation in the inputs of groups of perceptrons. In the MLP, the convolution operation is performed by the convolutional layers and afterwards, some activations must be selected by some pooling layers. By taking this 2D image motivation, it is reasonable to imagine a CNN with only 1 dimension. These are called 1D-CNNs or known as temporal CNNs, since they are great with timeseries forecasting.

Let us formally define a 1D-CNN. A 1D-CNN takes as input a 3-dimensional tensor of shape (b, T, F_0), where b is the batch size, T is the length of each timeseries, and F_0 is the number of timeseries used [17]. Throughout every layer, b and n keep constant, but F_l might increase or decrease in each layer l towards the output layer. For simplicity, we will disregard batches, so $b = 1$. Let $X_t \in \mathcal{R}_0^F$ be the input tensor of features vector, that has a length of F_0 features, for time step $1 < t < T$. Visually, this corresponds to a vertical column of the input tensor in Figure 2. In a convolutional layer l, a 1D filter, F_l is applied that captures how the input features evolve over the course of time. Given L convolutional layers, a filter is applied to each (equal or not) [17]. The filters for each layer, l, are parametrized by the weights explained above, $\mathbf{w}^{(l)} \in \mathcal{R}^{F_l \times k \times F_{l-1}}$, where k is the kernel size (or filter duration), and a bias, $b_0^{(l)} \in \mathcal{R}^{F_l}$. For the l-th convolutional layer, the i-th component of the activation, $\hat{\mathbf{a}}_t^{(l)} \in \mathcal{R}^{F_l}$, is a function of the incoming activation matrix, $\mathbf{a}_t^{(l-1)} \in \mathcal{R}^{F_{l-1} \times T_{l-1}}$, from the previous layer [17]:

$$\hat{\mathbf{a}}_{i,t}^{(l)} = g\left(b_i^{(l)} + \sum_{t'=1}^{k} \langle \mathbf{w}_{i,t',.}^{(l)}, \hat{\mathbf{E}}_{.,t+k-t'}^{(l-1)} \rangle\right) \tag{7}$$

where g is an activation function, such as the ReLu function. Notice the convolution operation iterating kernel by kernel window of size k. Essentially, each component i of the output vector of each layer is given by taking the dot product of the sub-sequence of the input and a kernel vector of learned weights of the same length. To get the next *i+1* component of the output, the kernel window is shifted *to the right* by one and the same procedure is applied. One can already see how the temporal evolution is being taken into account.

Followed by each convolutional layer, a pooling layer should be applied, which basically *downsamples* the timeseries. The pooling operation has a pool size, p, and for each sub-sequence of p components of the previous convolutional layer output, it chooses the component with maximum activation value (MaxPool); or

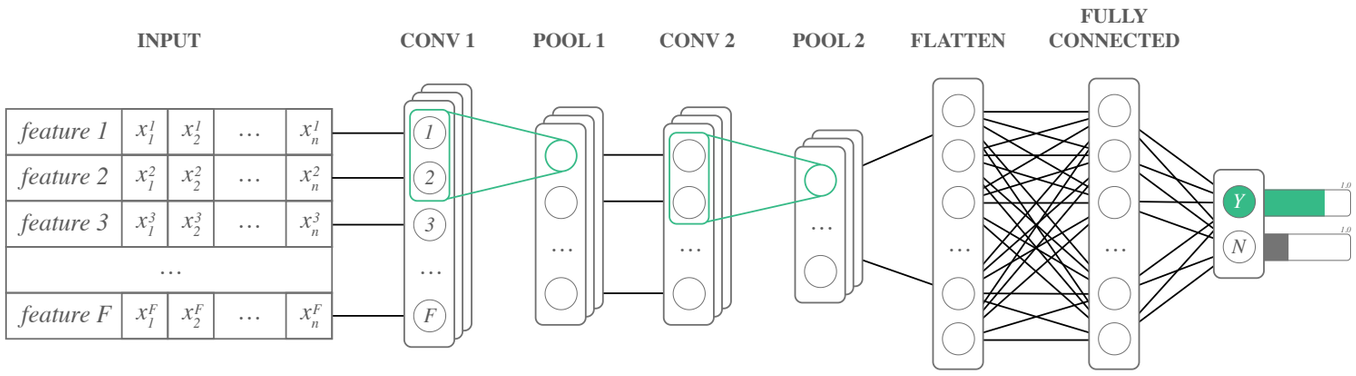

Fig. 2. Suggestive representation of one of our CNN models. Multiple, let's say F, 1-dimensional timeseries of features of length n can be fed at the input layer, which will have a shape of $(n \times F)$. The scheme presents two pairs of convolutional-pooling layers, inspired by **??**, followed by flattening layer and a fully connected layer for structural classification. The perceptron with highest probability at the end of an epoch (or test), is the classification done by the network, namely "(Y)es it is a seizure", or "(N)o it is not a seizure". Normalization and dropout layers are not shown.

takes the average of the p activation values (AvgPool). So, the timeseries length are transformed as $T^{(l)} = \frac{1}{p}T^{(l-1)}$ [17]. If $p = 2$, timeseries decrease by half.

At the end, the-1D CNN also returns a 3-dimensional tensor, which can latter be fed into a flattening layer to be used in a *regular* fully-connected MLP, as described in the beginning [17].

Regarding the training algorithms, many can be used such as the classic stochastic gradient descent (SGD), or its contemporary adaptations like the Adam and AdaDelta, which theoretically speaking promise better results. The AdaDelta [18] was proposed in 2012 with the advantage of having an adaptive learning rate. Instead of accumulating all past squared gradients (activations), Adadelta restricts the window of accumulated past gradients to a fixed size w, by summing the previous gradients recursively, and saving them as a decaying average of all past squared gradients. Hence, the current average $E_t[g^2]$ at time step t only depends on the average of the previous time step and current gradient:

$$E_t[g^2] = \gamma \cdot E_{t-1}[g^2] + (1-\gamma) \cdot g_t^2 \tag{8}$$

where γ is a decay constant. The update function in terms of classical SGD notation, gives the update of each time step as:

$$\Delta_t = -\frac{\eta}{\sqrt{E_t[g^2] + \epsilon}} \cdot g_t \tag{9}$$

where η is the learning rate and ϵ is an error constant.

...

III. State of the Art

In this section we summarize the related work being done by other groups with similar tasks. We divided it into two sections: (A) detection of crisis onset and (B) prediction of crisis in the pre-ictal period.

A. Detection of Epileptic Seizures

See a more thorough revision of the detection classifiers currently being used in Table II. Below in text we highlight just a few key groups.

1) Support Vector Machines:

Behbahani *et al.* developed a classification algorithm based on SVMs to classify epileptic and non-epileptic signals based on 5 minutes segments of ECG recording. A total of 170 seizures were analyzed, with 86 seizures having a left-sided focus and 84 seizures having a right-sided focus. The HRV signals were analyzed using EPILAB (a MATLAB toolbox), and time domain (mean NN intervals, mean HR), frequency domain (LF, HF, LF/HF ratio) and non-linear features (SD1, SD2, and SD2/SD1 ratio) were extracted. These features were then assessed using a least squares SVM (LSSVM) classifier coupled with a Leave-One-Out Cross-Validation (LOOCV) approach, used to demonstrate the consistency of the classification results. The best classification results provided a sensitivity of 83.13% and 76.47%, a specificity of 90.36% and 82.35%, and an accuracy of 86.74% and 79.41%, in right and left-sided focus seizures, respectively [19].

2) Feed-Forward Neural Networks:

The same group of Behbahani *et al.* proposed algorithms based on neural networks that successfully classified the non-epileptic and epileptic segments using HRV data. Ten features were extracted from the data, including mean HR, maximum beat per minute (MBPM) and mean NNI for the time domain analysis; LF, HF and LF/HF ratio for the frequency domain analysis; and standard deviation of instantaneous beat-to-beat NNI variability (SD1), standard deviation of continuous long-term NNI variability (SD2), short-term variability to long-term variability ratio (SD1/SD2), and area of the ellipse fitted to the Poincaré plot (S). Extracted features were used as the input for MLPs with different number of hidden layers and five training algorithms (Resilient back propagation – RP, Conjugate gradient with Powell–Beale restarts – CGB, Polak–Ribière conjugate gradient – CGP, One-step secant quasi-Newton – OSS, and Levenberg–Marquardt – LM) were designed. The LM training algorithm returned the best values for the evaluation metrics used, obtaining a sensitivity, specificity and accuracy of 88.66%, 90% and 88.33%, respectively, in secondary generalized seizures, and 83.33%, 86.11% and 84.72%, respectively, in complex partial seizures [20].

3) Convolutional Neural Networks:

Successful results have been achieved by some groups when detecting arrhythmia types and other non-typical ECG waveforms with CNN models [21]–[23]. In 2020, Hsieh *et al.* were able to detect arterial fibrillation (AF) events with 1D convolutional neural networks (1D-CNN). Their experimental results showed an average F1 score of 78.2% [21]. They fed the models directly with post-processed ECG recordings. Their architecture consisted of 10 convolutional layers extracting features and the 3 fully connected layers to structurally classify each input timeseries into "AF" or "non-AF" classes. Their training with the Adam optimizer

TABLE I
SELECTED FEATURES FOR EACH PATIENT

Patient ID	MeanNN	SDNN	RMSSD	NN50	pNN50	Var	HR	MaxHR	LF	HF	LF/HF	HF/LF	SD1	SD2	CSI	CSV	REC	DET	LMAX	S	KFD	SampEn	CoSEn	Total
101	•	•				•	•		•	•	•	•			•	•	•							11
102	•	•					•	•		•	•	•				•	•					•		10
103	•	•	•			•	•	•		•		•			•	•	•	•		•			•	14
106	•	•	•			•	•		•	•	•				•	•		•	•	•				13
107	•	•	•				•	•		•		•			•	•	•						•	11
108	•	•	•			•	•	•							•	•	•					•	•	11
109	•					•	•		•	•		•			•	•	•	•		•			•	12
Total	7	6	4	0	0	5	7	4	3	6	3	5	0	0	6	7	6	3	1	3	0	2	4	

consisted in about 40 epochs, with cross-validation, and a base learning rate of 0.001.

In 2017, Zhang *et al.* were able to engineer 1D-CNN architectures to classify four different types of arrhythmia and to discriminate them from non-arrhythmia patterns. Their CNN was much more shallow with only two convolutional layers and one fully connected layer. Their most optimized CNN model learns effective features and completes classification automatically with an accuracy of 97.5%. To our knowledge this was the most successful and robust method to date. Inspired by these works, we will try to test such models to detect ANS changes in the ECG extracted features.

B. Prediction of Epileptic Seizures

As aforementioned, epileptic crises are often associated with modifications in the ANS, which usually precede the onset of seizures of several minutes. Thus, there is a great interest in identifying these modifications enough time in advance to prevent a dangerous effect and to intervene.

Billeci *et al.* developed patient-specific SVM classifiers to predict seizures using features from NNI series. After a preprocessing step and the reconstruction and correction of the NNI series, time domain (mean NNI, variance NNI, RMSSD, SDNN, NN50, PNN50), frequency domain features (LF, HF, LF/HF), nonlinear features (SD1, SD2, SD2/SD1 ratio, CSV, CoSEn, KFD), as well as features obtained from recurrence quantification analysis (RQA) (%REC: Recurrence Rate, %DET: Determinism, LMAX, LAM: Laminarity, TT: Trapping Time, ENT: Entropy) were extracted. Each vector of features was segmented into three different states: ictal, beginning at seizure onset and lasting approximately 3 minutes; preictal, starting 15 minutes prior to the seizure onset; and interictal, containing all the data preceding the preictal state and/or proceeding the postictal state. A feature selection step was then performed in order to identify the most relevant features for discriminating between preictal and interictal phases. Afterwards, the data was classified into these two states using a cost sensitive SVM to balance out the datasets, obtaining an average sensibility of 89.06%, with a number of false positive per hour (FP/h) of 0.41. A second experiment was performed to predict unseen seizures on the basis of previous ones, using a double-cross-validation approach in patients who had at least 3 seizures. The results were dependent on seizure type, with mean accuracy, sensitivity, specificity and FP/h values of 74.62%, 70.08%, 76.98% and 3.36, respectively, being obtained for all seizures/patients. An average prediction time of 13.7 minutes was achieved [24].

In the work developed by Sargo *et al.*, morphological features were employed for the prediction of epileptic seizures based on ECG data with promising results. For morphological features, the group chose the mean Euclidian distance beat, mean Euclidean distance QRS, mean Pearson correlation beat and QRS. For each seizure crisis, the samples within 30 minutes before the onset were labelled as the pre-ictal class, and the samples occurring in the first 30 minutes of the session with a duration of 30 minutes were labelled as the inter-ictal class. The supervised classification algorithms used in this work were k-NN (K-Nearest Neighbours), SVC (Soft-Margin Support Vector Machines with radial basis kernel), and GaussNB (Gaussian Naive Bayes), which achieved a maximum F1-score of 67%, 68% and 68%, respectively, for some patient-crises pairs [25].

Fujiwara *et al.* proposed a new HRV-based epileptic seizure prediction method consisting of HRV feature extraction from NNI data of epileptic patients, and epileptic seizure prediction by using multivariate statistical process control (MSPC), an anomaly monitoring technique. Eight features were extracted from the NNI data (mean NNI; SDNN; RMSSD; Total Power (TP); LF; HF; LF/HF). The prediction algorithm was able to discriminate the patient status between preictal and interictal. The data 15 minutes before and 5 minutes after seizure onsets were said to be preictal episodes. The data recorded during periods apart at least 50 minutes from seizure onsets were defined as interictal episodes. Their results showed that almost all features, in particular frequency domain features, changed before the seizure onset, while the interictal HRV features did not fluctuate so much. An overall sensitivity of 91% was obtained when applying the prediction method [26].

...

IV. RESULTS AND DISCUSSION

In the feature selection step, a patient-specific approach was taken. Overall, 10 to a maximum of 14 features were selected with an average of 11.7 features per patient (Table I). The most frequently selected features were: meanNN, SDNN, Var, HR, HF and HF/LF, CSI, CSV and REC.

A. Classification with SVM

After feature selection, tunning of the SVM hyperparameters was performed for each patient. For all patients, the optimal kernel function was the radial basis function (RBF) kernel, with gamma equal to 1:

$$K(\mathbf{x}, \mathbf{x}_i) = e^{-\gamma||\mathbf{x}-\mathbf{x}_i||}, \quad \gamma = 1.0 \tag{10}$$

where $\mathbf{x}_i$ is the input sample, $\mathbf{x}$ is the space of the training set, and $||\cdot||$ represents the Euclidean distance operator The C parameter ranged between 10 and 50, depending on the patient. The C works as a penalty parameter for the error term, that controls how the optimizer should avoid misclassifications in each training example. For large values of C, the optimizer will choose a smaller-margin hyperplane if that hyperplane *does a better job* of getting all the training points classified correctly.

For the classification, a patient specific approach was chosen, since patients presented different manifestations in the various features, as well as different manifestation times, with some having clear changes on the preictal state, while others only presented changes in the postictal state. The classification results obtained by the trained SVM on the test set for each patient are reported in Table III. The SVM classifier was successfully used in the classification of different types of epileptic seizures from various patients, obtaining an average F1-score, accuracy,

TABLE II

SUMMARY OF THE CLASSIFICATION ALGORITHMS USED FOR THE DETECTION OF EPILEPTIC SEIZURES, ACCORDING TO THE LITERATURE.

Seizures Detected	**Segments Size**	**Feature extraction**	**Feature selection**	**Detection algorithm**	**Cross-validation**	**Results**	**References**
21 seizures in 8 newborns. Seizure type not specified	64-s non-overlapping ECG epochs (21 seizure related and 13 non seizure related)	time domain (mean, SD, Hjorth parameters: activity, mobility, complexity) and time-frequency features	Yes. two-phase wrapper-based feature selection technique	linear classifier, quadratic classifier, and k-nearest neighbor (kNN), where k = 1, 3, and 5	Leave One Out Cross-Validation (LOOCV)	best results being obtained for the 1-NN classifier. The proposed algorithm achieved 85.7% sensitivity and 84.6% specificity.	[28]
patient-independent, automatic detection of seizures in 14 newborns. The mean seizure length in this study is 3.83 min	non-overlapping 60 s epochs	Sixty-two features. Time-domain (Mean NN, SDNN, CV, NN temp, SDNN temp, CV temp, RMSSD,pNN5, pNN10 pNN15, pNN20, pNN25, SDSD, delNN, maxdiff, SD1, SD2, CSI, CVI, del plus, Hsh, del minus, A(T) 5, A(T) 10, A(T) 15 A(T) 20, A(T) 25, Llen, NE, ZC) and frequency-domain (Power in VLF, LF and HF bands, Hs , Psub) features	Yes. Based on the weights from the linear SVM	the HR-based SVM with linear kernel, the HR-based SVM with RBF kernel and the SVM with RBF kernel evaluated on a subset of features	Fivefold cross-validation	The performance of all three HR-based systems was seen to be inconsistent across patients and within patients. On evaluating the system using multiple patients an average ROC area of 0.59 with sensitivity of 60% and specificity of 60%, were obtained.	[29]
6 seizures experienced by 6 patients with temporal lobe epilepsy. Three seizure types: GTCS, CPS, SPS	64 R-R intervals with maximum overlapping	Frequency domain features (LF, HF, LF/HF, LF/(LF+HF) and reciprocal HF-power	No	Statistical analysis	No	Reciprocal power peaks from 10 s preictal to 24 s postictal were 2.96- 93.63 higher than in controls.	[30]
170 seizures analyzed (86 seizures with a left-sided focus, 84 seizures with a right-sided focus)	5 minutes	Time domain (mean RR intervals, mean HR), frequency domain (LF, HF, LF/HF ratio) and non-linear features (SD1, SD2, and SD2/SD1 ratio)	No	least squares SVM (LSSVM)	LOOCV	The best classification results provided a sensitivity of 83.13% and 76.47%, a specificity of 90.36% and 82.35%, and an accuracy of 86.74% and 79.41%, in right and left-sided focus seizures, respectively	[19]
Partial Epilepsy	30-32 s	Frequency domain (mean of absolute deviation of fast Fourier transform coefficients, and spectral entropy) and time domain features (activity, mobility, complexity)	No	Threshold approach wherein certain thresholds are set for each feature and seizure and non-seizure conditions are classified accordingly; SVM algorithm.	No	Threshold approach was found better for seizure detection. The classification result revealed 94.2% accuracy, 84.1% sensitivity, and 94.5% specificity for frequency domain analysis and 96.6% accuracy, 78.7% sensitivity, and 97.2% specificity for time domain analysis. The average latency by the threshold approach was also lower , obtaining values between 0-35 s.	[27]
206 seizures was collected from 15 patients. Seizure types: CP, SG	-	Ten features were extracted from the data, including mean HRs, maximum beat per minute (MBPM) and mean RR interval for the time domain analysis; LF, HF and LF/HF ration for the frequency domain analysis; and SD1, SD2, SD1/SD2 ratio, and area of the ellipse fitted to the Poincare´ plot (S).	No	Multilayer perceptron neural networks (MLP) with different number of hidden layers and five training algorithms (Resilient back propagation – RP, Conjugate gradient with Powell–Beale restarts-CGB, Polak –Ribière conjugate gradient- CGP, One-step secant quasi-Newton- OSS, Levenberg–Marquardt- LM)	No	The LM training algorithm returned the best values for the evaluation metrics used, obtaining a sensitivity, specificity and accuracy of 88.66%, 90% and 88.33%, respectively, in SG seizures, and 83.33%, 86.11% and 84.72%, respectively, in CP seizures	[31]

TABLE III
SVM CLASSIFICATION RESULTS ON THE TEST SET.

Patient ID	Accuracy (%)	Sensitivity (%)	Specificity (%)	F1-score (%)	FP (%)
101	90.1	64.3	96.6	69.1	3.3
102	-	-	-	72.5	4.1
103	96.8	84.1	99.2	74.0	0.8
106	97.8	92.2	99.1	76.6	0.9
107	94.9	88.1	97.4	80.5	2.6
108	93.0	68.7	97.4	66.6	2.6
109	97.3	93.3	98.1	90.8	1.9

sensitivity, specificity and FP of 75.6%, 94.9%, 81.8%, 97.9% and 2.6%, respectively (Table III). These values are similar to the ones obtained in the literature when using SVM algorithms for epileptic seizure detection [19], [24], [27].

Following the methodology of [24], we also performed some tests on the patient-specific models with samples from other crises episodes that were not used in the fitting-testing of the model – called by them as *unseen crises*. For instance, the classifier for patient 102 was trained by 80% of samples of crises 1, 2 and 3 and tested with the remaining 20% of samples of the same crises. Then an *unseen crisis* number 4 was used in the second round of tests. For that round, we got an average F1-score, accuracy, sensitivity, specificity and FP of 59.2%, 72.3%, 16.5%, 87.8%, 8.8%, respectively (Table IV). These values are much worse than the ones obtained on the test set partitioned upon the fitting and evaluation of the models. This may indicate slight overfitting by the trained models to the tested data of the same crises episodes. The steepest decline occurs for the sensitivity value, which is to be expected, since our data is imbalanced, with an extremely limited number of true positives. The accuracy and specificity remain high, since most of the data is correctly classified as true negatives, with very few false positives happening.

These values could be improved by training with a larger volume of data (higher number of crises per patient), as well as using as the interictal period random data from the baseline, eliminating the issue related to feature alterations already present before the preictal state and into the postictal state. This could allow a greater discrimination between the two periods, increasing the evaluation scores. This is confirmed when the baselines of the patients are fed to the trained models, and all data points are classified as true negatives. To increase model robustness, we could also choose lower SVM hyperparameters, C and γ. While relatively high C and γ values may lead to higher training evaluation scores, it also may cause worse performance on *unseen crises*.

Furthermore, a preliminary decision algorithm was implemented to provide a YES or NO output given features of a crisis/baseline as input, by using the SVM models previously trained. A good accuracy score was achieved, with all inputs being correctly classified. However, to note that a very small dataset was used to test the algorithm. Also, the accuracy was also not tested in terms of pinpointing exactly the onset of the crisis, or the beginning of the preictal state, which would be necessary when applying it with end-users, so it remains to implement that.

B. Classification with CNN

Inspired by the work of [21]–[23] we tried to use temporal convolutional neural networks (1D-CNN) with the same feature vectors as for SVMs to investigate whether they could be used in the future to detect epileptic crises. Here the premise is different: we are not trying to find a maximum margin separator in a higher dimensional space, but instead we are trying train classifiers with the temporal evolution of each feature waveform, in the hope that if future unseen crises follow the same temporal pattern, then they should be classified as seizures. In the other hand, if they follow the temporal pattern of the baseline signals, they should be classified as not seizures. This is very similar to what our brain does when trying to detect patterns in the feature waveforms of Figure 3.

We explored different timeseries duration for the features vectors, and the one that we got any admissible results was 15 minutes before onset and 2 minutes after onset – making a total of 17 minutes. After preparing the datasets, with normalized features, we performed a comparative analysis in patient 102, 103 and 108. The one that got good results was only patient 108. Regarding different architectures, we present in Table V how the different configurations can influence the classifier accuracy. Figure 2 shows a schematic representation of a CNN with 2 hidden convolutional-pooling layer pairs (more can be added), and a fully connected layer before the output layer.

First of all, we can notice from Table V how the feature selection process impacts the classifier accuracy. The trained model could only correctly predict the seizure segments when a more restrict set of 8 features was selected, instead of using all 23 features as input. This makes the input shape reduce for almost 1/3, which will further reduce the complexity of future layers. Complex data is usually not a problem for neural networks, but data must be statically significant and, preferentially, not redundant. The same feature selection step was applied as for SVMs, and only 8 relevant features were identified for patient 108, among all its four crises: mean and variance NNI, SDNN, RMSSD, Max-HR, CSI, CSV, Sample entropy and respective coefficient of the NNI signal. Each timeseries had a total of 123 samples ($\approx$ 17 minutes), since the features were extracted with segments of 15 seconds duration and 5 seconds overlap between each segment pair. Cross-validation was used where in each fold, only one crisis would be saved for testing (see Methods). For the dataset reserved for fitting, 80% was used for training and 20% for validation of each epoch.

The effect of increasing the number of hidden layer pairs *confuses* the network, leading to lower accuracies. This might be happening for three reasons: i) the remaining of the model architecture is not well configured when shifting to deeper networks; or ii) the deeper the network, the more examples needed to train it (as a rule of thumb), which we do not have; or iii) deeper networks

TABLE IV
SVM CLASSIFICATION RESULTS ON *unseen crises* TEST SET.

Patient ID	Accuracy (%)	Sensitivity (%)	Specificity (%)	F1-score (%)	FP (%)
102	84.1	28.2	95.0	59.0	5.0
106	57.4	28.1	71.8	60.2	28.2
	74.3	3.4	96.5	71.6	3.4
107	-	-	-	64.1	0.7
	-	-	-	61.1	4.4
108	73.3	6.25	87.7	39.3	12.3

TABLE V
SEIZURE DETECTION ACCURACY FOR DIFFERENT CNN ARCHITECTURES

Features	Input shape	# Hidden layers	# Filters	Kernel shape	Activation function	Dropout Hidden	Pool size	# Dense units	Training algorithm	Correctly predicts seizure	Correctly predicts baseline
All	(123, 23)	2	64	3	*relu*	0.5	2	100	Adam	No	1/10
			16			0.2		32		No	1/2
Selected sdnn, rmssd, mean, max-hr, csi, csv, var, sampen, cosen	(123, 8)		18	7			5	128	Adadelta	Yes	3/5
			22	7						Yes	3/4
		3	32	9						Yes	1/2
		4	64	9		0.3		(32, 64)		No	1/4
		5	128	7		0.4				No	1/5
										No	1/5

simply might not work with this kind of features and might not be a good solution.

Regarding the kernel shape, we believe that increasing it will lead to better results, but only up to a maximum. We found out a kernel of size 7 (7×1) to be the most indicated. In practice this means we are performing convolution operations in the features of each segment of 45 seconds of the original NNI signal. The pool size of each subsequent pooling layer was also found to yield better results if increased. Maximum pooling operations were used always, since average pooling did not result in good accuracies.

The fully connected (dense) layer at the end of the networks, before the output layer, is also extremely important to structurally classify the time evolution of the features after convolution. Only one layer of 32, 64 or 128 perceptrons seems to accurately classify the test seizure and most of the baseline input tests. Conversely to what was expected, by increasing the number of fully connected layers or perceptrons, no matter how, the classifier performance decreases. This might be again due to one of the three reasons pointed out for the convolutional layers. To support that, we know that like all statistical models, neural networks are subject to overfitting when there are too many parameters in the model [16], and we do not have many examples to counter-balance that.

The best performance was achieved with a model trained by Adadelta, composed of two convolutional hidden layers of kernel size 7 and 18 filters in both, and a maximum pooling of size 5 subsequent to both of these. Normal padding is added to the convolutional layers and they are activated by the *ReLu* function. Other functions such as the hyperbolic tangent were tried, but with no success. Also, no stride was found needed, since the features were already extracted with overlapping. Additionally, a batch normalization layer was added after the first convolutional layer, which reestablishes normalization after each training epoch, so that weights changes do not amplify along the training process. Additionally, after each pair of convolutional-pooling layers, a dropout layer was inserted that randomly drops out 20% of the previous units, hence offering stochastic stability between training epochs. Along with that, the model performs better with a fully connected layer of 128 perceptrons at the end.

...

V. CONCLUDING REMARKS

SVMs are currently one of the golden techniques for seizure prediction and detection. In this work, SVM models were successfully used in the classification of epileptic seizures, using a patient-specific approach. Employing the trained SVM models on unseen crises yielded worse results, possibly indicating a slight overfitting of the models. For future work, more patient data should be used to train the models in order to obtain a more accurate and robust SVM classification algorithm. It would also be interesting to fit and evaluate models based on the type of crisis, independent of patient.

CNN results were also promising but more work should be conducted to validate their use for this application. Temporal convolution takes into consideration the temporal evolution of the signal/features, which other models like the SVMs do not. We suggest that networks more deep should be tried in the original ECG signal, since these ones were very shallow, violating then the whole concept of a deep neural network. Also, by using the post-processed ECG signal, features could be extracted directly from the original timeseries with the convolutional layers, and the fully connected layers should be used for structural classification, having it all together in just one unified model.

...

METHODS

The ECG signals were acquired at Hospital de Santa Maria (Lisboa) as part of a current research project outside of our own. Each crises was firstly identified by the patient or caregiver, by pressing a button, and later confirmed by a medical doctor specialized in epilepsy by interpreting the EEG signal that was being acquired simultaneously, after signal processing. The pre-processing consisted in conditioning the signal with a FIR band-pass filter of order 150, with a band from 5 Hz to 20 Hz. Faulty segments were corrected by comparison and substitution with the average template. The Hamilton segmenter [32] was used to detect the R peaks of the QRS complexes. The Hamilton segmenter *forces* a distance of 200 ms for all large peaks and a distance of 360 ms for all confirmed R peaks. It works with the time elapsed since the last detection and the average NNI of the last 8 detected R-peaks. That is, the detection threshold is a function of the average noise and the average QRS peak values of the last 8 detected R-peaks. The NN intervals (NNI) were computed by the distance between consecutive R peaks. Afterwards, the NNI timeseries was upsampled from 1 to 4 Hz.

Data from a total of 7 patients was analysed, with varying ages and seizure types (Table VI). Of the total number of crises presented by each patient, some crises ended up being excluded due to them being electrical crises, with little to no manifestation, and due to poor quality signal/acquisition issues.

For each patient, the classifiers results were assessed using accuracy, sensitivity, specificity, F1-score, and the false positive (FP) rate metrics, defined by the following equations.

$$Accuracy(\%) = 100 \cdot \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$Sensitivity(\%) = 100 \cdot \frac{TP}{TP + FN} \tag{12}$$

$$Specificity(\%) = 100 \cdot \frac{TN}{TN + FP} \tag{13}$$

$$F1 - score(\%) = 100 \cdot \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{14}$$

$$FP(\%) = 100 \cdot \frac{FP}{FP + TN} \tag{15}$$

where TP, TN, FP, FN mean true positive (correctly classified preictal + ictal segments), true negative (correctly classified interictal segments), false positive and false negative, respectively. A higher relevance was given to the F1 score since it better reflects the performance of classification on imbalanced data.

We divided our software analysis in a pipeline of four main stages:

1) Feature Extraction
2) Feature Selection
3) Classification
4) Decision

Inside our code package `/src` directory, there can be found two modules named `feature_extraction` and `feature_selection`, corresponding respectively to the first two stages of the pipeline. There can also be found the modules `svm` and `cnn` corresponding to the SVM classification models and to the CNN classification models, respectively. And module `decision` contains the preliminary decision algorithm. All software was developed and tested in Python 3.8 and it is wrapped on a virtual environment that includes the following main libraries: h5py, numpy, pandas, scipy, sklearn, tensorflow.keras, and PyQt5. Next we briefly explain how the code is organized.

A. Feature Extraction module

This module has one important auxiliary script called `io.py` that is responsible for reading and writing HDF5 files from-to the `/data` directory. In `/data/patients.json`, it can also be found a descriptive file holding every patient and crisis metadata used in this work. This `io.py` is also responsible from retrieving metadata from it and this one-way abstraction should not be broken in other code parts.

A superclass to extract HRV features was created called `HRVFeaturesCalculator` and multiple calculators can be derived from this superclass. The goal is to keep groups of related features together and separated from others not related. Each *calculator* must receive an NNI signal to work on. Six specialized *calculator* classes were developed by us, although some code was already given to us:

- `TimeFeaturesCalculator`
- `FrequencyFeaturesCalculator`
- `PointecareFeaturesCalculator`
- `KatzFeaturesCalculator`
- `RQAFeaturesCalculator`
- `COSenFeaturesCalculator`

As their names indicate before the suffix `FeaturesCalculator`, each of these is meant to compute a group of features. Some are linear features, others are not. The `TimeFeaturesCalculator` can extract the time-domain features, namely the mean, the variance, the SDNN and the RMSSD of the given NNI signal. It can also extract the features NN50, pNN50, Mean HR and maximum beats per minute. The `TimeFeaturesCalculator` can extract the frequency-domain features, namely the low frequency power (LF), the high frequency power (HF), and their ratios. The `PointecareFeaturesCalculator` can extract features from the Poincaré plot, namely the SD1, SD2, CSI, CSV, and area of the ellipse fitted to Poincaré plot (S). The `KatzFeaturesCalculator` can extract the katz fractal dimension. The `RQAFeaturesCalculator` can extract features from recursive quantitative analysis, namely the REC, the determinant, and the maximum L. Finally, `COSenFeaturesCalculator` can extract the sample entropy and the coefficient of sample entropy.

Note that this architecture allows for future extensibility with no effort. If one wishes to add a feature to a calculator, all one has to do is to write a *getter* function named after the feature keyword, e.g. `get_myfeature`, that extracts and returns the feature from the NNI signal, and add a readable label in `labels` associated to it, e.g. `myfeature: My Feature Label`.

The main file of the module contains useful functions to be called to extract and save features. The more complete and robust one is `extract_patient_hrv_features`, which allows to extract some or all features from a patient set of crises or all crises. A patient number and a crisis number (or a list of crises numbers) should be given. If no crisis is given, it extracts features for all crises of that patient. The time of each segment should be given in seconds, and also the overlap between each segment, if needed. Regarding the features, we can select one or more groups of features to be extracted. For instance, if we want to extract only time-domain features, we shall only mark `_time` as `True`. Also, a list of individual features might be extracted – they must be explicit by their keywords in `needed_features`. The algorithm computes features resultant from the union between the groups of features marked as `True` and the individual features specified. Hence, a personalized combination can be achieved. Moreover, `baseline` should be marked as `True` to extract features from the baseline signal as well. In that case, specify `state` as "`awake`" or "`asleep`". In the end, a total of 21 features were extracted for each patient using segmentations of 15 seconds with an overlap of 5 seconds.

B. Feature Selection Module

The main file of this module allows the researcher to observe on the graphical user interface (GUI) a table with all the features extracted from a set of pairs patient-crises, and also an image plotting the features of the same set of pairs, grouped by features. So, each plot shows the extracted feature vector of all pairs patient-crises. The vectors of all pairs are aligned to a single crisis onset timepoint (marked in red). A median tendency line is also drawn in each plot (optional). This allows for a thorough visual inspection of the features correlation, in order to see which are a better fit for a specific classifier. The features are shown normalized and with outliers already removed. Figure 3 shows an example of three selected features for patient 108 in all its four crises, but the visualization supports up to 23 features, at least.

Feature normalization and outlier removal functions was already provided to us and can be found in `utils_signal_processing.py`. Normalization can

TABLE VI
CHARACTERISTICS OF THE PATIENTS, WITH CORRESPONDING NUMBER OF SEIZURES, SEIZURE TYPE, LOCALIZATION AND LATERALIZATION.

Patient ID	Age (years)	Number of seizures	State	Type of Seizure*	Localization**	Lateralization	Number of crises removed
101	19	3	Asleep	FBTC	FT, CP	Left	-
102	54	6	Awake/Asleep	FWIA(4)/E(2)	T	Left	2
103	40	6	Awake	FWIA(5)/FBTC(1)	F, T,CP, FC	Left(4)/Bilateral(2)	3
106	36	3	Awake	FWIA	PO,O	Left	-
107	28	50	Asleep	E(28)/FUNK(9)/F(2)	F, T,CP	Right(40)/Bilateral(6)	33
108	45	4	Awake	FWIA	T, PT	Right	-
109	15	8	Awake	F(1)/FBTC(4)	T,TP	Left	6

*FBTC-Focal to bilateral tonic clonic; FWIA- Focal with impaired awareness; E- Electric;FUNK- Funk focal unknown mental status; F-Focal without impaired awareness.

**F- Frontal; T-Temporal; O-Occipital; FT-Fronto-temporal; CP- Centro-parietal; FC- Fronto-central; PO- Parieto-occipital; PT-Parieto-temporal; TP-Temporo-parietal.

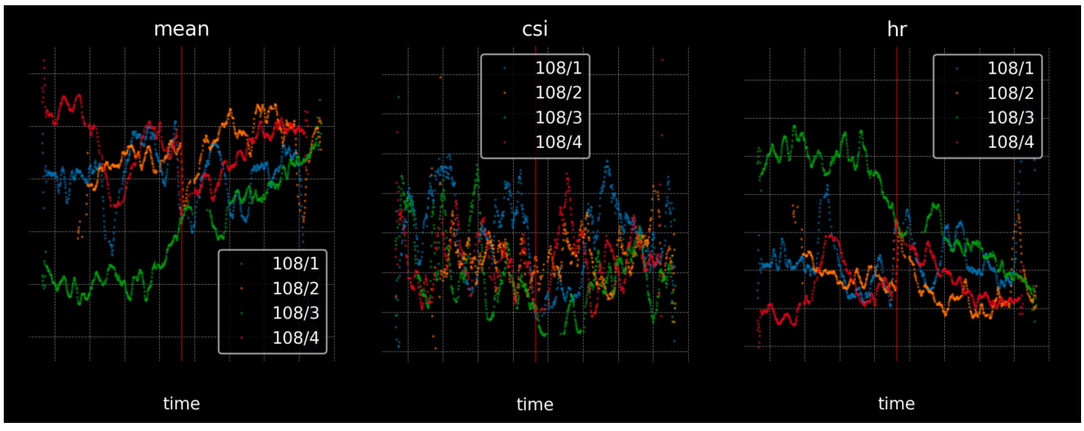

Fig. 3. Feature Selection plots for visual inspection. Example of only 3 features of patient 108 and all its four crises. Each pair patient-crisis is colour-coded. All episodes are aligned to a reference crisis onset, marked by a red vertical line.

be performed using a minimum-maximum approach or in a normal distribution approach. Outliers may be removed by filtering the samples with lowest z-scores, or by removing the samples below quantil 1 and above quantil 3.

C. SVM module

This module applies a SVM to train and test our data. The functionality is implemented in the procedure `train_test_svm`. First, `create_dataset` is used to perform a segmentation of the feature vector into 3 main states: preictal, from 20 minutes prior to the seizure onset to the seizure onset; ictal, from seizure onset to the end of seizures (2 minute seizure duration was assumed); and interictal, non-seizure data preceding the preictal state or proceeding the postictal state. The states are then divided into two classes: class 0 consisting of interictal state, and class 1 consisting preictal plus ictal states. The duration of the preictal state and the ictal state are taken as inputs. After segmenting the dataset, the data is separated into train and test set using `train_test_split` from scikit-learn. The test set size is given as input as a fraction of the full dataset size. This split takes into account the stratification of the data. For each patient, we randomly selected 80% of the dataset for training, and evaluated the classification by testing the model on 20% of the dataset.

For the SVM model, an additional feature selection step was performed. Two different wrapper techniques were tested, backward elimination (`backwards_elimination`) and recursive feature elimination (`rfe`), with the backward elimination algorithm providing the most accurate results. In this algorithm we start by choosing a criterion for our model, in this case, a p-value > 0.05. We then fit an Ordinary Least Squares (OLS) regression model including all the features. Then, the feature whose loss gives the most statistically insignificant deterioration of the model fit is deleted (higher p-value), and this process continues until no further variables can be deleted without a statistically insignificant loss of fit.

SVMs use a set of hyperparameters that can be optimized for model training, namely the regularization parameter, C, a kernel function, and kernel coefficients, such as a γ for RBF, or a $degree$ in the case of a polynomial kernel. The procedure `hyperparameter_tuning` searches for the best combinations of hyperparameters out of the possible values that we define in the `param_grid` variable, by using `GridSearchCV` from scikit-learn. In the hyperparameter tuning we applied a 10-fold cross validation, using as metric the F1-score, and the class weights were taken into account. The best parameters are saved and can be used in future runs of the program.

After finding the best parameters, the model is trained using a `fit` from scikit-learn and 10-fold stratified cross validation. A classification report is printed for the train set, with the F1-score also being calculated. Finally, `predict` from scikit-learn is used for the classification of the test set. A confusion matrix is also drawn, from each we extract the accuracy, sensitivity, specificity, the F1-score and the FP rate.

The `train_test_svm` procedure is implemented inside a wrapper, `train_test_svm_wrapper`, which performs a selected number of iterations. This allows for the results to be uncoupled from the initial train test split. In this work, we opted by performing 10 iterations of each experiment and averaging the scores. After we are satisfied with the trained model, we can then save it in pickle format.

D. Decision module

In the decision module we implemented a rudimentary algorithm for a YES or NO threshold approach to the presence of crises (`decision_algorithm`). The best SVM model and previously selected features for a given patient should be first loaded. The model is then used to obtain a prediction. If at any point, two successive inputs are said to belong to class 1, the algorithm detects a crisis. If it fails to find two ones in a row, it detects there are no crises for that segment.

E. CNN module

This module is divided into three files representing each step of the methodology: the `load_dataset`, which prepares a dataset in a correct format to be taken as input in a CNN model; the `create_model`, which creates a CNN model with the given characteristics; and the `fit_evaluate`, which takes a CNN model, fits and evaluates it with a given train set and a test set.

The `prepare_dataset` procedure receives a patient number and state (”`awake`” / ”`asleep`”). The inputs temporal interval is going to be x minutes before each crisis onset (specify x in `before_onset_minutes`), plus y minutes after each crisis onset (specify y in `crisis_minutes`). The procedure uses all crises (n) of the given patient and, due to the limited number of examples, it is hard-coded to use only 1 crisis to the test set – all others are used for training. One should indicate the number of which crisis should be used for testing in `test_crisis`. The procedure prepares the same number of training baseline inputs as the number of training crises inputs ($n-1$). It selects arbitrary segments from the baseline signal of that patient. There is an option to add more baseline inputs to the testing set if one would like to better test the resilience of the models to false positives and true negatives. How many baseline tests should be specified in `n_baseline_tests`. This imbalance does not interfere in the training process, of course since it only adds more arbitrary

baseline tests to the testing set. Unfortunately, we cannot add more crises tests. The procedure allows for preparing inputs out of *pure* NNI signals or from their extracted features (`raw=False`). If to prepare features, as we did, one should specify them by their keywords in the `feature_inputs` list. Moreover, the number of dimensions of the model in which the prepared sets are meant to be used should be specified in `dimensions` – e.g. to prepare datasets for a 1D CNN, give `dimensions=1`. Essentially, this procedure will return:

- Set of train inputs
- Set of train targets
- Set of test inputs
- Set of test targets
- Inputs shape
- Targets shape

Procedure `create_model` uses `tensorflow` to create and return CNN models with the given specified architecture. Firstly, a convolution dimension should be specified in `dimension`, and an input shape must be given by `input_shape` (get this from `prepare_dataset`). If a fraction of input units should be dropped out, specify it in `dropout_visible`.

Then, there can be two types of hidden layers: convolutional layers and pooling layers. The number of hidden convolutional layers is deducted from the length of `filters` and `kernels`, where we should specify the number of filters in each layer, and the kernel shape in each layer, as lists, respectively. In the same way, the activation function, stride, padding, dilation, L1 regularization, and L2 regularization of each hidden convolutional layer can be specified in the respective arguments. If batch normalization should be performed after the first convolutional layer, pass `batch_normalization` as `True`.

For each convolutional layer, a pooling layer follows, which can take the maximum activation (`pool_type = 'max'`) or the average activation (`pool_type = 'avg'`) of the previous convolutional layer. Specify the number of units of the pooling operation in `pool_size`. After every pair of convolutional-pooling layers, if a fraction of units should be dropped out, specify it in `dropout_hidden`.

Afterwards a flatting layer is applied and a sequence of fully connected layers follows. The number of perceptrons of each of these layers should be given as list in `fully_connected_layer_size`. Its length defines the number of layers. For each of these layers, if a fraction of perceptrons should be dropped out, specify it in `dropout_fc`. Finally, an output layer follows with the number of classification classes given by `output_size`, with an activation function given by `output_activation_function`.

All this dynamic procedure allows us to have a factory of CNN models to easily create models with different architectures, making comparative analysis simple and less cumbersome.

Procedure `fit_and_evaluate_model` receives a created CNN model, regardless of its architecture, a train set of inputs and one of targets, a test set of inputs and one of targets, and fits (trains + validates) the model with the inputs for as many epochs as specified in `epochs`, or if convergence is reached according to `patience`. Convergence is said to be reached if for `patience` number of epochs, the validation accuracy has not increased. The fraction of inputs for validation, that are taken from the test input set, must be specified in `validation_split`. Different optimizer algorithms, loss functions, and metrics may be defined in the respective arguments. After fitting, the procedure takes the test inputs, tries to predict outputs for them, compares them with he test targets and yields an accuracy and loss.

Procedure `do_experiment` allows us to experiment a model and investigate if it is stable, by performing `fit_and_evaluate_model` ten times and yielding the average accuracy and average loss at the end of the experiment. The main file of the module shows an example of preparing a dataset, creating a model, and running an experiment. To test this on different patients, segments, or architectures, one should only change the necessary parameters accordingly.

...

Code and Data Availability

The developed software is publicly available at https://github.com/jomy-kk/epilepsy_mlb. The data used, namely the NNI signals, is not available to share, as it is not property of the group.

Acknowledgments

Thank you to Mariana Abreu who helped us with insightful technical knowledge, provided us with the datasets and some signal processing code snippets.

References

[1] R. S. Fisher, C. Acevedo, A. Arzimanoglou, A. Bogacz, J. H. Cross, C. E. Elger, J. Engel, L. Forsgren, J. A. French, M. Glynn, D. C. Hesdorffer, B. Lee, G. W. Mathern, S. L. Moshé, E. Perucca, I. E. Scheffer, T. Tomson, M. Watanabe, and S. Wiebe, "ILAE Official Report: A practical clinical definition of epilepsy," *Epilepsia*, vol. 55, no. 4, pp. 475–482, Apr. 2014. [Online]. Available: http://doi.wiley.com/10.1111/epi.12550

[2] O. Devinsky, A. Vezzani, T. J. O'Brien, N. Jette, I. E. Scheffer, M. de Curtis, and P. Perucca, "Epilepsy," *Nature Reviews Disease Primers*, vol. 4, no. 1, p. 18024, Jun. 2018. [Online]. Available: http://www.nature.com/articles/nrdp201824

[3] G. J. Tortora and S. Nammi, *Introduction to the human body (10th ed.)*. Milton: John Wiley & Sons, 2014, oCLC: 964408650. [Online]. Available: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4745420

[4] A. Pitkänen, P. S. Buckmaster, A. S. Galanopoulou, and S. L. Moshé, Eds., *Models of seizures and epilepsy*, second edition ed. London: Elsevier/Academic Press, 2017, oCLC: ocn965353713.

[5] S. Cui, L. Duan, Y. Qiao, and Y. Xiao, "Learning EEG synchronization patterns for epileptic seizure prediction using bag-of-wave features," *Journal of Ambient Intelligence and Humanized Computing*, Sep. 2018. [Online]. Available: http://link.springer.com/10.1007/s12652-018-1000-3

[6] M. Avoli, "A brief history on the oscillating roles of thalamus and cortex in absence seizures," *Epilepsia*, vol. 53, no. 5, pp. 779–789, May 2012.

[7] E. Tsakiridou, L. Bertollini, M. de Curtis, G. Avanzini, and H. C. Pape, "Selective increase in T-type calcium conductance of reticular thalamic neurons in a rat model of absence epilepsy," *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, vol. 15, no. 4, pp. 3110–3117, Apr. 1995.

[8] E. Sitnikova and G. van Luijtelaar, "Electroencephalographic characterization of spike-wave discharges in cortex and thalamus in WAG/Rij rats," *Epilepsia*, vol. 48, no. 12, pp. 2296–2311, Dec. 2007.

[9] R. S. Fisher, J. H. Cross, J. A. French, N. Higurashi, E. Hirsch, F. E. Jansen, L. Lagae, S. L. Moshé, J. Peltola, E. Roulet Perez, I. E. Scheffer, and S. M. Zuberi, "Operational classification of seizure types by the International League Against Epilepsy: Position Paper of the ILAE Commission for Classification and Terminology," *Epilepsia*, vol. 58, no. 4, pp. 522–530, Apr. 2017.

[10] M. K. Smyk, I. V. Sysoev, M. V. Sysoeva, G. van Luijtelaar, and W. H. Drinkenburg, "Can absence seizures be predicted by vigilance states?: Advanced analysis of sleep–wake states and spike–wave discharges' occurrence in rats," *Epilepsy & Behavior*, vol. 96, pp. 200–209, Jul. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1525505019300484

[11] S. Vieluf, R. El Atrache, S. Hammond, F. M. Touserkani, T. Loddenkemper, and C. Reinsberger, "Peripheral multimodal monitoring of ANS changes related to epilepsy," *Epilepsy & Behavior*, vol. 96, pp. 69–79, Jul. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1525505018309223

[12] S. Behbahani, "A review of significant research on epileptic seizure detection and prediction using heart rate variability," *Turk Kardiyol Dern Ars*, vol. 46, no. 5, pp. 414–421, 2018.

[13] F. Shaffer and J. P. Ginsberg, "An overview of heart rate variability metrics and norms," *Front Public Health*, vol. 5, no. 258, 2017.

[14] S. Behbahani, N. J. Dabanloo, and A. Nasrabadi, "Ictal heart rate variability assessment with focus on secondary generalized and complex partial epileptic seizures," *Advances in Bioresearch*, vol. 4, no. 1, pp. 50–58, 2013.

[15] F. Leutmezer, C. S. S. Lurger, K. Pötzelberger, and C. Baumgartner, "Electrocardiographic changes at the onset of epileptic seizures," *Epilepsia*, vol. 44, no. 3, pp. 348–354, 2003.

[16] S. J. Russell, P. Norvig, E. Davis, and D. Edwards, *Artificial intelligence: a modern approach*, third edition, global edition ed., ser. Prentice Hall series in artificial intelligence. Boston Columbus Indianapolis New York San Francisco Upper Saddle River Amsterdam, Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo: Pearson, 2016.

[17] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks: A Unified Approach to Action Segmentation," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, vol. 9915, pp. 47–54, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-49409-8_7

[18] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[19] S. Behbahani, N. J. Dabanloob, A. M. Nasrabadic, and A. Dourado, "Classification of ictal and seizure-free hrv signals with focus on lateralization of epilepsy," *Technology and Health Care*, vol. 24, no. 1, pp. 43–56, 2016.

[20] S. Behbahani, N. J. Dabanloo, A. M. Nasrabadi, C. A. Teixeira, and A. Dourado, "A new algorithm for detection of epileptic seizures based on hrv signal," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 26, no. 2, pp. 251–265, 2014. [Online]. Available: https://doi.org/10.1080/0952813X.2013.861874

[21] C.-H. Hsieh, Y.-S. Li, B.-J. Hwang, and C.-H. Hsiao, "Detection of Atrial Fibrillation Using 1D Convolutional Neural Network," *Sensors*, vol. 20, no. 7, p. 2136, Apr. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/7/2136

[22] L. Niu, C. Chen, H. Liu, S. Zhou, and M. Shu, "A Deep-Learning Approach to ECG Classification Based on Adversarial Domain Adaptation," *Healthcare*, vol. 8, no. 4, p. 437, Oct. 2020. [Online]. Available: https://www.mdpi.com/2227-9032/8/4/437

[23] D. Li, J. Zhang, Q. Zhang, and X. Wei, "Classification of ECG signals based on 1D convolution neural network," in *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*. Dalian: IEEE, Oct. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8210784/

[24] L. Billeci, D. Marino, L. Insana, G. Vatti, and M. Varanini, "Patient-specific seizure prediction based on heart rate variability an recurrence quantification analysis," *PLoS ONE*, vol. 13, no. 9, 2018.

[25] F. Sargo, A. Fred, H. Silva, and C. Bentes, "Analysis of electrocardiographic patterns for epileptic seizure prediction," 2019.

[26] K. Fujiwara, M. Miyajima, T. Yamakawa, E. Abe, Y. Suzuki, Y. Sawada, M. Kano, T. Maehara, K. Ohta, T. Sasai-Sakuma, T. Sasano, M. Matsuura, and E. Matsushima, "Epileptic seizure prediction based on multivariate statistical process control of heart rate variability features," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 63, no. 6, pp. 1321–1322, 2016.

[27] G. Shamim, Y. U. Khan, M. Sarfraz, and O. Farooq, "Epileptic seizure detection using heart rate variability," 2016.

[28] M. B. Malarvili and M. Mesbah, "Newborn seizure detection based on heart rate variability," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 11, pp. 2594–2603, 2009.

[29] O. Doyle, A. Temko, W. Marnane, G. Lightbody, and G. Boylan, "Heart rate based automatic seizure detection in the newborn," *Medical engineering physics*, vol. 32, no. 8, pp. 829–839, 2010.

[30] J. Jeppesen, S. Beniczky, A. Fuglsang-Frederiksen, P. Sidenius, and Y. Jasemian, "Detection of epileptic-seizures by means of power spectrum analysis of heart rate variability: A pilot study," *Technology and Health Care*, vol. 18, pp. 417–426, 2010.

[31] S. Behbahani, N. J. Dabanloob, A. M. Nasrabadic, and C. T. amd Antonio Dourado, "A new algorithm for detection of epileptic seizures based on hrv signal," *Journal of Experimental Theoretical Artificial Intelligence*, vol. 26, no. 2, pp. 251–265, 2014.

[32] P. Hamilton, "Open source ECG analysis," in *Computers in Cardiology*. Memphis, TN, USA: IEEE, 2002, pp. 101–104. [Online]. Available: http://ieeexplore.ieee.org/document/1166717/