POLITECNICO DI TORINO

Master degree course in Embedded Systems

Master Degree Thesis

One-shot Learning for Seizure Detection and Identification of Epileptogenic Brain Regions from Long-time Human iEEG Recording with End-to-end Binary Operations

Supervisors:
prof. Andrea Calimera
prof. Luca Benini

Candidates
Alessio Burrello
number: 238495

University Tutor
doct. Abbas Rahimi

October 2018

Summary

This thesis presents an efficient algorithm by combining symbolic dynamics and brain-inspired hyperdimensional (HD) computing for both seizure onset detection and identification of ictogenic (= seizure generating) brain regions from intracranial electroencephalography (iEEG). Moreover, the simplicity of the algorithm eases its implementation on an embedded AI computing device platform (e.g. the NVIDIA Jetson TX2 Module) for long term operation.

The proposed algorithm provides: (1) a unified method for both learning and classification tasks with end-to-end binary operations; (2) one-shot learning from seizure examples; (3) linear computational scalability to any number of electrodes; (4) generation of transparent codes with interpretable features; (5) a simple embedded implementation which is fast and energy efficient.

The algorithm first transforms iEEG time series from each electrode into symbolic local binary pattern codes from which a distributed representation of the brain state of interest is constructed across all the electrodes and over time in a hyperdimensional space. Such *holographic* representation is used to quickly learn from seizures, detect their onset, and identify the spatial brain regions that generated them. Moreover, HD computing is characterized by one-shot or anyway fast learning, making it a prime candidate for utilization in such a domain with a typical low quantity of training data.

I assess the performance of the proposed algorithm on two different

dataset: (1) the first contains 99 short-time iEEG recordings from 16 drug-resistant epilepsy patients being implanted with 36 to 100 electrodes; (2) the second is composed by 18 long-time recordings from 18 drug-resistant epilepsy patients: a total of 2656 interictal hours and 120 seizures are contained in the recordings. All the patients come from the epilepsy surgery program of the Inselspital Bern.

On the first dataset, for the majority of the patients (10 out of 16), our algorithm quickly learns from one or two seizures and perfectly (100%) generalizes on novel seizures using k-fold cross-validation. For the remaining six patients, the algorithm requires three to six seizures for learning. Our algorithm surpasses the state-of-the-art including deep learning algorithms by achieving higher specificity (94.84% vs. 94.77%) and macroaveraging accuracy (95.42% vs. 94.96%), and 74× lower memory footprint, but slightly higher average latency in detection (15.9 s vs. 14.7 s).

On the second dataset,the algorithm learns from one or two seizures and achieves 0.0 false detection rate for all the patients. The state-of-the-art achieves again lower latency in detection (12.8 s vs. 17.3 s), but higher false detection rate (0.31 f/h)

Moreover, the algorithm can reliably identify (with a p-value < 0.01) the relevant electrodes covering an ictogenic brain region at two levels of granularity: cerebral hemispheres and lobes.

Finally, the algorithm shows 15× gain in execution time and 18× gain in energy consumption with respect to the state-of-the-art competitors, when implemented on the NVIDIA TX2 platform.

Acknowledgements

I would like to thanks my advisor Abbas Rahimi for its support and its ideas throughout the development of my project. I also thank Kaspar Schindler for providing the datasets, giving us the opportunity to contribute to this very interesting field. Finally, I thank the IIS department of ETH for providing the material and infrastructure to conduct this thesis, and Politecnico of Turin for the funds to move to Zurich.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Epilepsy is a severe and prevalent chronic neurological disorder affecting 0.6–0.8% of the world's population [1]. One third of epilepsy patients continue to suffer from seizures despite best possible pharmacological treatment [2]. For these patients with so-called drug-resistant epilepsy [3] efficient algorithms are urgently required to detect the onset of seizures and ultimately identify the ictogenic (i.e. seizure-generating) brain regions for possible surgical removal [4, 5].

This procedure has to be done extremely accurately, because the duration of intracranial EEG recordings is limited—typically to 1 to 3 weeks [6]—to minimize the discomfort for the patient. Thus there are most often only a few seizures recorded in the epilepsy monitoring unit.

Different alternatives have been explored in the past [7], using movement based systems [8] or heart rate [9]. Despite the presence of many techniques, intracranial electroencephalography (iEEG) currently provides the best spatial resolution and the highest signal-to-noise ratio (SNR) of electrical brain activity recordings.

Recent studies have shown successful application of machine learning methods [10–15] using iEEG signals to detect two distinct states of brain

activity in patients with epilepsy, i.e., interictal (= between seizures) and ictal (= during seizures). These methods are based on extracting useful features followed by traditional supervised machine learning methods (such as support vector machines [11,14], Bayesian analysis [15], and artificial neural networks [11]), and more recently deep learning algorithms [12,13]. These methods are however seriously challenged by the need to reliably detect seizures from a small number of examples. This is due to the patient-specific nature of seizure dynamics (i.e. activity patterns at onset, propagation and termination of seizures), and to the inherent asymmetry in the iEEG recording, namely that the ratio of interictal to ictal segments is typically very large [14,16].

In addition, these conventional methods face other important challenges including:

1. The outcome of their learning is often a "black box" that is not *transparent* to an expert neurologist, hence cannot be analyzed for better diagnosis, e.g., preciseley delineating the ictogenic brain regions.

2. Their high computational complexity and memory demands render them unsuitable for real-time detection on resource-limited wearable or implantable devices.

3. Their offline and slow (iterative) training time prevent them from online and incremental learning from new seizure occurrences, hence they cannot be quickly adapted to new dynamics.

4. They operate with few electrodes, e.g. 6 [15], 16 [14], and 22 [13] electrodes. However, a larger number of electrodes is mandatory to properly assess the spatio-temporal evolution and spreading of epileptic seizures [5,17,18] and to properly identify the borders of the seizure onset zone for surgical resection [19].

   Furthermore, recent studies have demonstrated that iEEG recordings

from outside the seizure onset zone provide important information about seizure generalization [20] and thus might be helpful to prevent its occurrence and thereby decrease the risk for sudden unexcpected death [21].

Despite the impressive results shown ($\tilde{1}$00% accuracy) from some of these works [11,12], they are tested on the same dataset [22]. The dataset is composed by 500 segments of 23.6 seconds of single channel recording, extracted from ictal or interictal periods of 3 epilepsy patients and 2 healty people. This dataset shows important limitations:

1. single channel recordings, i.e. I consider only the channels nearer to the seizure onset, neglecting all the farer ones: however, the location of the Seizure Onset Zone is not always available before the starting of the alghoritm training;

2. the 23.6 seconds segments are totally inside interictal or ictal periods, losing the natural transition between the two states;

3. the time covered from 100 interictal segments is $\tilde{3}$9 min, which is too low to assess the specificity of a real time working algorithm: 99% of Specificity involves more than 1 false detection/hours.

In this work I will introduce two novel dataset to better test the seizure detection algorithms, with multi channel recording and many hours of recording.

One promising option is computing with simple linear binary codes to avoid otherwise expensive operations such as costly floating-point arithmetic. Combining methods from symbolic dynamics and information theory is a computationally efficient approach. At its core it consists of analyzing the occurrence of patterns and even bears similarity to classical visual EEG interpretation [23].

In this thesis, I propose a new algorithm to address the aforementioned challenges by the following contributions:

1. **I propose a single algorithm for both learning and classification tasks by jointly exploiting symbolic dynamics and brain-inspired vector-symbolic architectures.** The proposed algorithm combines methods from symbolic dynamics [23–25] (Section 1.2.2) and brain-inspired computing [26] (Section 1.1) that supports one-shot or few-shot learning, i.e. the ability to learn object categories from one or few examples. Symbolic dynamics models a dynamical system by a discrete space consisting of sequences of abstract symbols, each of which corresponds to a state of the system. At the heart of this new algorithm is a brain-inspired vector-symbolic computational theory called hyperdimensional (HD) computing [26] that learns quickly by computing with random vectors in a very high dimensionality, also referred to as *hypervectors*. My proposed algorithm consists of analyzing the occurrence of symbols or patterns—that even bears similarity to classical visual EEG interpretation [23]—followed by one-/few-shot learning. First, as an elegant *symbolization* method, I exploit local binary patterns (LBP) [25] to map a sequence of iEEG samples into a small bit string as a symbol. Second, these symbols are projected into an HD space that enables reliably combining them over time and across electrodes to encode a compact representation (i.e., a prototype vector) for one-shot learning from the state of interest. Further, I use the same algorithm for both learning and classification tasks: the algorithm initially learns from few ictal or few interictal segments by writing the corresponding prototype vector (ictal or interictal) into an associative memory, and then classifies new segments based on Hamming distance among these two *learned* prototype vectors.

2. **The algorithm is simple, computationally scalable, and operates with end-to-end binary operations:** (i) For every iEEG electrode, the LBP feature extractor directly transforms the time series into symbols as bit strings with limited length. (ii) HD computing

then projects the symbols to an HD space and computes a distributed long binary vector that encodes occurrences of the symbols among all electrodes (an approximation method to encode histograms of symbols). (iii) The training and classification are performed by simply bundling and comparing the binary vectors. (iv) The classification decision is followed by a patient-dependent voting to reduce the false alarms. Further, the computational complexity of the algorithm linearly scales for any number of input electrodes. This scalability provides a universal interface to homogeneously cover all patients with different numbers of implanted electrodes (e.g. 36 to 100) and seizure dynamics. The concurrent use of LBP and HD computing enables end-to-end execution of my algorithm with simple binary codes to avoid otherwise expensive operations such as costly floating-point arithmetic (Section 2.1).

3. **One-shot learning and comparison during short time recordings.** I provide a dataset from 16 drug-resistant epilepsy patients that contains 99 iEEG recordings, each one consisting of a 3 minutes interictal (i.e. immediately pre-ictal) segment and the ictal segment followed by a 3 minutes postictal segment (Section 4.1). Using this dataset, I compare my algorithm with the state-of-the-art methods using the LPB with a linear support vector machine, or a fully connected neural network [11], as well as deep learning algorithms [12, 13] (Section 4.2.1). My algorithm quickly learns from one seizure (for eight patients), or two seizures (for two more patients), and perfectly (100%) generalizes on detecting novel seizures with k-fold cross-validation. For the remaining six patients, the algorithm requires 3–6 seizure examples for learning. Overall, my algorithm surpasses the state-of-the-art methods: compared to [12], it achieves higher specificity (94.84% vs. 94.77%) and macroaveraging accuracy (95.42% vs. 94.96%), and

a 74× lower memory footprint. My algorithm has slightly higher average latency in detection (15.9 s vs. 14.7 s), but it can raise an alarm in the first 8% of the ictal window. We also provide the public access to the code of my algorithm and the anonymized dataset with links provided in the papers that will follow.

4. **One-shot learning and comparison during long time recordings.** I provide a second dataset with 2656 hours of recording from 18 drug-resistant epilepsy patients that contains 120 seizures. A single recording for each patient is provided, with marked begin and end of the seizures. Using this second dataset, I provide the same comparisons in terms of sensitivity, false detection x hour and delay (Section 4.2.2). My algorithm quickly learns from one seizure or two seizures, and achieves an ideal 0.0 false detections on $\tilde{3}000$ hours of interictal period. Overall, my algorithm surpasses the state-of-the-art methods: compared to SVM [11], it achieves lower false detections (0.0 f/h vs. 0.32 f/h) and higher sensitivity (87.7% vs. 83.5%), but higher latency in detection (17.2 s vs. 12.8 s).

5. **My algorithm produces transparent codes for identifying seizure-generating brain regions.** Due to the well-defined set of arithmetic operations with inverses in HD computing, the learned prototype vectors—i.e. the binary codes derived from the iEEG recordings during the ictal and interictal brain states—are transparent and analyzable with interpretable features. My algorithm identifies the ictogenic brain regions by measuring the relative distances between the learned prototypes that are produced from different electrodes (Section 2.2). Such identification is done at two levels of spatial resoultion, the cerebral hemispheres and lobes, with p-value < 0.01 (Section 4.3). This takes the application of my algorithm beyond the traditional scope of seizure onset detection by automatically identifying ictogenic

> brain regions that can provide more accurate data to better target surgical resection and thus potentially improve post-surgical seizure control. It enables post-translational support for clinical decision making, and is in sharp contrast to those machine learning methods that only produce "black boxes."

Furthermore, the algorithm is truly scalable and provides a simple interface (with minimal number of parameters) to universally operate with all patients having 24 to 128 electrodes implanted.

Finally, I show the ease of implementation of my algorithm on an embedded platform, the NVIDIA Jetson TX2. The algorithm has been written using CUDA Toolkit 8.0 GA1 together with C code, to fully parallelize the binary operations on the GPU of the TX2. The algorithm has been tested on the TX2 platform, and compared to the state-of-the-art-methods in terms of time and energy: the HD implementation is 15× faster than competitors and achieve 17× gain in energy consumption for the classification of a window of 0.5 seconds.

1.1 Hyperdimensional Computing

The human brain consists of billions of neurons, glial cells, and synapses, suggesting that large circuits are fundamental to its computational power. Hyperdimensional (HD) computing [26] explores this idea by computing with random vectors in a very high dimensionality (d), also referred to as *hypervectors*. To represent basic items, or symbols, HD computing starts by selecting a set of *atomic* vectors: d-dimensional (pseudo)random vectors with independent and identically distributed (i.i.d.) components. This thus conforms to a *holographic* or *holistic* representation: the encoded information is distributed *equally* over all the d components such that no component is more responsible to store any piece of information than another hence maximizing robustness. When the dimensionality is in the thousands, e.g.

d=10,000, it yields a huge number of nearly orthogonal atomic vectors (see Section 1.1.1). This lets HD computing to combine two atomic vectors into a new *complex* fixed-width vector using well-defined vector-space operations (Section 1.1.2), while keeping the information of the two atomic vectors with high probability [27]. Hence, it is also called *holographic reduced representations* (HRRs) meaning that the reduced (fixed-width) descriptions have less information about components than the full descriptions [28] This overall provides a novel perspective on data representations and associated operations with unique features in terms of robustness and speed of learning [29–37].

Learning and classification with HD computing is composed of three main steps:

1. mapping symbols to *atomic* high-dimensional vectors;
2. combining atomic vectors with well-defined arithmetic operations in an encoder to produce composite structural vectors;
3. storing/updating (i.e., learning) these vectors inside an associative memory and finally comparing with query vectors (i.e., inference).

HD computing begins with selecting a set of random high-dimensional vectors (with i.i.d. components) to represent basic objects. They serve as atomic vectors and are used as building blocks to construct representations of more complex objects. To generate these atomic vectors, I use random d-dimensional vectors of equally probable 1s and 0s, i.e., dense binary elements of $\{0,1\}^d$. Nevertheless, bipolar components, i.e. $+1$ and -1 could be used with the same results; in this work I will use always dense vectors, i.e. with 50% distribution of 1 and 0, with respect to sparse vectors, where number of 1 is lower than number of 0 [38].

These vectors are stored to a so-called item memory (IM), i.e. a symbol table or dictionary of vectors defined in the system. In my seizure detection

system, the names of electrodes and the LBP codes are the basic symbols. The IM assigns a random orthogonal vector to every symbol.

Here, I focus on two main operations of HD computing for encoding with the atomic vectors: bundling and binding. Bundling, or addition, of binary vectors $[A + B + \ldots]$ is defined as the componentwise majority with ties broken at random. Binding is defined as the componentwise XOR ($\oplus$). Both operations produce a d-bit vector with an important distinction: bundling produces a vector that is *similar* to the input vectors, whereas binding produces a *dissimilar* vector.
Hence, bundling is well suited for representing sets, and can combine field/value bindings to produce a larger structure (e.g., record or tuple). Representations of such composite structures are constructed directly from representations of the atomic vectors by applying these operations without requiring any learning for the encoder.

The output vector of the encoder is then fed into an associative memory (AM) for training and inference. During training the output vector of the encoder is stored in the AM as a *learned* pattern. During inference the output of the encoder is compared with the learned patterns. Comparison is based on a distance metric over the vector space. The AM uses Hamming distance, defined as the number of different components of two binary vectors. In the following I better explain the Hamming distance (Section 1.1.1) and the HD operations (Section 1.1.2)

1.1.1 Measure of Similarity

Let us consider d-dimensional binary random vectors[1] of equally probable 1s and 0s, i.e., dense binary elements of $\{0, 1\}^d$. Using this dense binary code (aka binary spatter code [39]), the similarity between two vectors is

[1] In the thesis, I use capitalized italic letters to indicate vectors that can also appear with a subscript and superscript.

defined by the Hamming distance as the number of components at which they differ. To express the distance on a real scale of 0 to 1, I divide the Hamming distance by d denoted as:

$$\Delta(X, Y) : \{0,1\}^d \times \{0,1\}^d \rightarrow [0,1]. \tag{1.1}$$

In high dimensions, e.g. $d \geq 1000$, most points are $d/2$ bits apart from each other, which yields a normalized Hamming distance of $\Delta \approx 0.5$, and stands for two nearly orthogonal vectors [40]. This stems from the binomial distribution for $p = 1/2$ and $n = d$, where $d/2$ is the mean. Correlated vectors yield $\Delta \approx 0$ whereas $\Delta \approx 1$ implies anti-correlation [26].

HD computing begins by randomly generating a set of atomic vectors that represent basic items in the cognitive system. The atomic vectors are nearly orthogonal to each other, and are stored in a so-called *item memory* (IM). The IM is like a symbol table or dictionary of the items defined in the system, and stays fixed throughout the computation. In my seizure detection system, the LBP codes and the names of electrodes are the basic items (or symbols) that are assigned to the atomic vectors. These atomic vectors, inside the IM, are used as building blocks from which more complex vectors are constructed. Such complex vectors stand for a concepts or percepts. For a complex vector is composite in nature, it can be very similar to other complex vectors with similar composition and structure [41].

1.1.2 Main operations

In this section all the operations that has been used to create composite hypervectors will be described [26].

- **Similarity**: is an operation that returns a scalar value from two vectors. It is based on the distance, *hamming distance* for the binary version of vectors, *cosine similarity* for bipolar vectors.

– **Cosine Similarity**: for bipolar vectors I adopt a slightly different metric of the one exposed in Section 1.1.1, based on the componentwise multiplication:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{D} A_i \times B_i}{\|A\| \times \|B\|}. \tag{1.2}$$

The codomain of the function is [−1,1], where −1 is linked to $A = not(B)$, 1 to $A = B$ and 0 to cosine similarity of two random orthogonal vectors.

- **Addition (Bundling)**: is an operation on two or more vectors that yield a vector: $S = [A + B + \cdots + C]$, where $[\dots]$ indicates the normalization of the sum, used to represent S in the binary space. The addition is done componentwise with ties broken at random. Taken a single component of S:

$$S_i = \sum_{i=0}^{n} V_i \tag{1.3}$$

and considering the bipolar case to ease the understanding, I can identify 3 cases:

$$S_i = \begin{cases} 1, & \text{if } \sum_{i=0}^{n}(V_i = 1) < \sum_{i=0}^{n}(V_i = -1) \\ 0, & \text{if } \sum_{i=0}^{n}(V_i = 1) = \sum_{i=0}^{n}(V_i = -1) \\ -1, & \text{if } \sum_{i=0}^{n}(V_i = 1) > \sum_{i=0}^{n}(V_i = -1) \end{cases} \tag{1.4}$$

Zeros are assigned pseudo-randomly to 1 or −1 to mantain the bipolar domain. The sum vector will be similar to all the added hypervectors.

Example 1 *For the addition of two elements is very simple to demonstrate: fixed the value of the first vector A, 50% of the components of B will be equals by probability to the one of A and the other 50% will generate ties: breaking them randomly, S will be 75% similar to A.*

The same holds for B.
Furthermore, the bundling is commutatibe and approximately invertible, but not associative.

Example 2 *There is a specific case in which this statement is not valid: if more than one half of the vectors added are equals, all the information about the other vectors is lost. In figure 1.1 is reported the cosine similarity for the sum of 3 hypervectors, showing in the left part the problem aforementioned.*

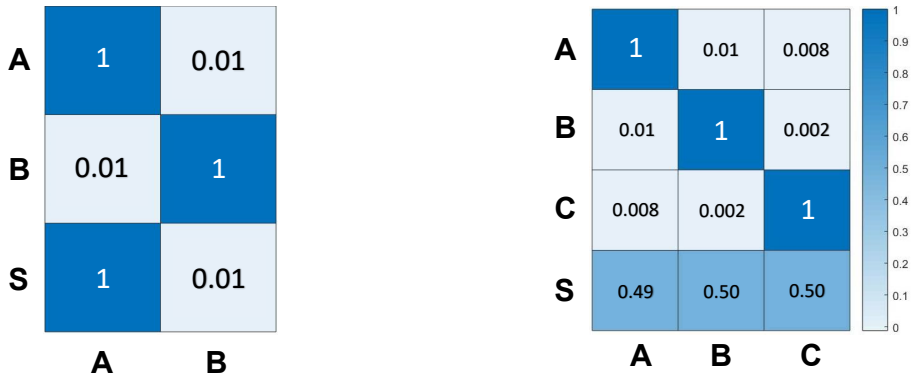

Figure 1.1: In each square, the respective cosine similarity is showed. The hypervectors A, B and C are generated randomly, with a cosine similarity $\simeq 0$. On the left, I have $S = [A + A + B]$, that will result in $S \equiv A$, because more than the half of hypervectors added are equal. On the right, $S = [A + B + C]$: the similarity between each component vector and the composite S is higher than the random (i.e. > 0).

- **Multiplication (Binding)**: is an operation on two vectors that yield a vector. $P = A \oplus B$ or $P = A \times B$, depending on the space in which I am working; both the operations are done componentwise. Conversely to the addition, it generates a third vector that is orthogonal to the

factors. The multiplication is normally used to link two objects in a composite structure. The most important properties are listed here (in the bipolar domain):

  - $I = A \times B$;
  - $P = A \times B \implies A = P \times B$, $B = P \times A$;
  - $C = A \times B + C \times D \implies C \times A = B + \text{noise} = B^1$, i.e. I can recover a noisy version of B in a composite vector, created with addition and multiplications.

- **Permutation**: is a unary operation on a vector that yield a vector, that is pseudo-orthogonal to the previous one, $B = \rho A$, where ρ is the rotation of one position of the vector. To obtain back A from B, I apply the inverse permutation. This operator is crucial when storing sequences, where I want to disitnguish a-b-c from b-a-c. Permutation preserves distances.

- **Normalization**: converts an intermediate result of an operation into an element of the space over which the operations are defined. If I work in the binary space, the addition of two vectors has to be normalized by a threshold function to make it binary; in the bipolar space, a simple sign operation is applied to return to the target space.

These three operations are combined to encode structures such as variable/value records, sequences, and sets.

1.1.3 Histogram Recall

I describe how the HD operations can be applied to encode histograms. The aim is to store an approximated version of a histogram in HD space to save memory and ease computation with only binary components. Below, I illustrate an example to better explain the procedure.

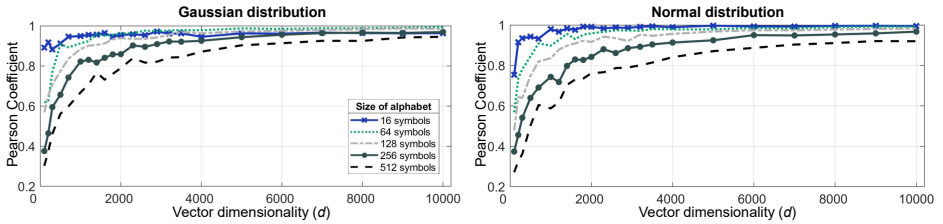

Figure 1.2: Pearson coefficient between exact histograms and their approximated HD versions. The procedure consists of the following steps: (1) creation of Gaussian and uniform distribution from an alphabet with m symbols where $m \in \{16,64,126,256,512\}$; (2) association of a random atomic vector to every symbol; (3) creation of the histogram vectors H for a window of 512 symbols by bundling their corresponding atomic vectors; (4) computing the similarities between H and the the atomic vectors; (5) Plotting Pearson correlation between similarities and the exact counts.

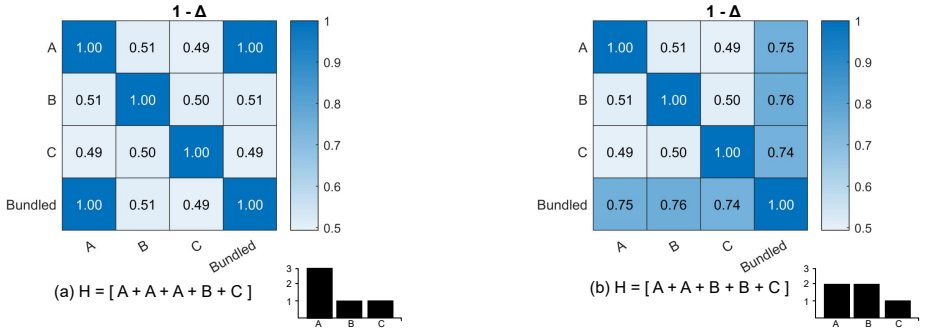

Figure 1.3: Similarity between three atomic vectors and two different encoded histograms. **(a)** symbol A occurs more than 50% of time, whereas **(b)** no symbol occurs more than 50% of time.

I use the LBP codes as basic symbols, and their counts in a 1 s window (i.e. 512 symbols) as a histogram. First, the LBP codes are mapped into atomic vectors through the IM. Then, the atomic vectors generated during the 1 s window are bundled to produce a complex vector (H) representing

the histogram. In this way, the histogram of LBP codes is *holistically* represented in a single binary vector H. The original LBP distribution can be recalled by comparing H with the individual atomic vectors. For every symbol, I compute the similarity as $1 - \Delta$ that recovers the count of the symbol. The set of the computed similarities represents the approximated histogram.

Fig. 1.2 shows the Pearson correlation coefficient between the similarity values (extracted from the approximated histograms) and the exact count of the symbols in the histograms. I consider five alphabets with different sizes of 16, 64, 128, 256, and 512 symbols. I use two different distributions to generate histograms from these alphabets for a window of 512 symbols: a Gaussian distribution to mimic polarized ictal histograms, and a uniform distribution to mimic randomly distributed interictal histograms. The experiments are repeated for vectors while varying the dimension (d). Using a $d > 2000$ with the 64-symbol alphabet, a Pearson correlation > 0.9 is observed for both distributions. The Pearson correlation further grows toward 1 with larger d. However, if a single symbol occurs more than 50% of the total symbols inside the histogram, the encoded H is a copy of that specific symbol (Fig. 1.3). This is due to the characteristics of the bundling operation (majority sum): if several copies of any vector are included in the bundling, then the resultant vector is closer to the dominating vector than to other vectors.

1.2 Encodings and preprocessing

The preprocessing stage is necessary in every machine learning algorithm. In fact, the data are almost never suitable for a classification problem. The data could be categorical, numerical, mixed or visual: in all the cases the data has to be encoded in the features space, that the learning algorithm will use to train.

In the case of a numerical signal pattern, as the iEEGs, you could learn sequence of raw numbers, or extrapolate informations from them; some typical futures are time and spatial correlation, frequency of the signal, power density, time reversability, and entropy of the signal.

In the next sections I will explain the different type of future extractor that my algorithms exploit.

1.2.1 Directed Horizontal Visibility Graphs

The directed horizontal visibility graphs (dHVG) [42] are used for mapping a time series such as iEEG signal into a network, well represented by the connection matrix. In figure 1.4 I illustrate the construction of the connection matrix for a short, artificially generated signal: it is composed by $y_{1,2,...7} = [5\ 2\ 6\ 5\ 4\ 6\ 7]$. In the visibility graphs domain, each point is mapped into a node: two nodes i and j are connected if there is no data point between them that is higher of y_i or y_j. Putting in a mathematical way I have:

$$n_{i,j} = \begin{cases} 1, & \text{if } min(y_i, y_j) > y_n \ \forall i < n < j \\ 0, & \text{if } \exists\ n \mid y_n > min(y_i, y_j) \end{cases} \tag{1.5}$$

As we can see from figure 1.4, the superdiagonal is always at one, since there are no point in between t_i and t_{i+1}.

This mapping technique has been demonstrated to be useful in the epilessy domain, to recognize epileptogenic brain regions [42], and to detect the seizure onset [42, 43]. In fact, this type of mapping perfectly encodes the time irreversability [44], pronounced in iEEG signals during the ictal stages.

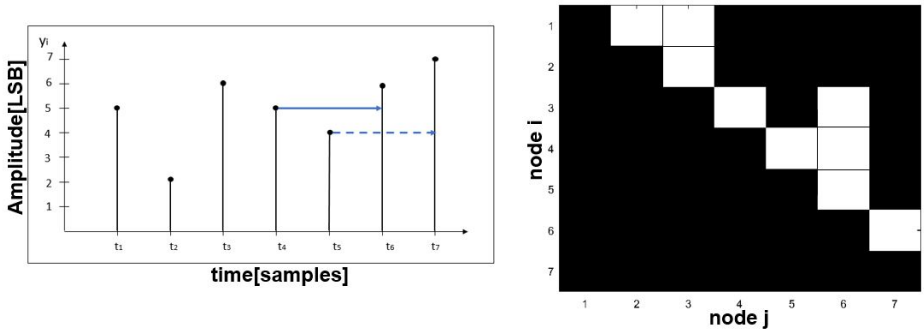

Figure 1.4: The procedure to construct the dHVG matrix is simple: let's take a single time sample and compare it to all the successive ones: if there is no points in between with value higher than the one of the two time samples under analysis, a link is drawn (continue raw in left graph); otherwise, a continue horizontal line can't be drawn between the two time samples, because there is an higher y_i in between. Hence, there is no link between the two points.

1.2.2 Local Binary Pattern Encoding

A class of data-analysis methods is referred to as symbolization, which describes the process of transforming raw experimental measurements into a series of discrete symbols.

Symbolization is particularly interesting for EEG analysis, because as recent experience has clearly demonstrated, it faithfully preserves dominant dynamical signal characteristics while significantly increasing the efficiency of detecting and quantifying information contained in real-world time series [45].
Symbolization may be efficiently achieved by mapping a sequence of iEEG samples into a bit string, i.e. a one-dimensional local binary pattern (LBP) [25]. A LBP code reflects relational aspects between consecutive values of the original iEEG signals only, but not the values themselves.

Computing a LBP code is simple:

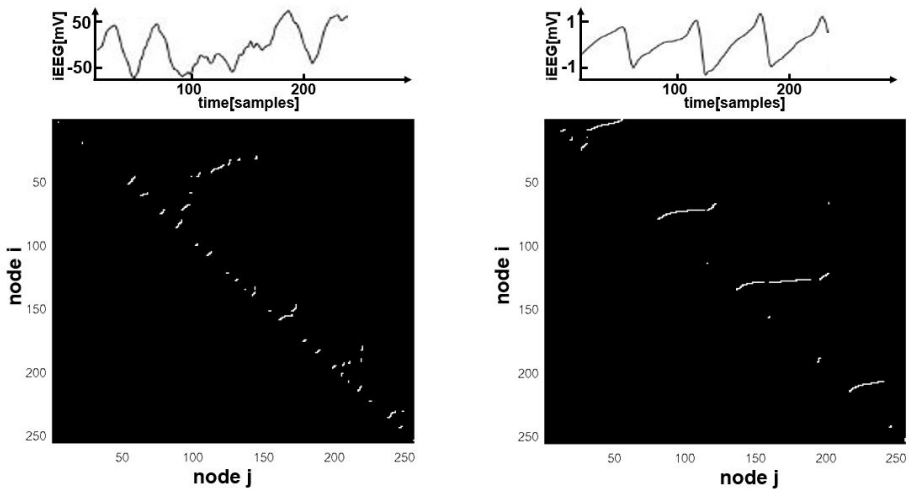

Figure 1.5: On the left the connection matrix for the interictal signal shown above, on the right for an ictal signal. In both the connection matrix the superdiagonals have been removed; in fact, since I want to compare the two connection matrix, is meaningless to keep a constant bias.

1. decide the number of points of the string of LBP (l);

2. the iEEG signal samples are converted into a bit string depending on the sign of the temporal difference of adjacent samples, $d_i = p_i - p_{i+1}$;

3. a LBP code of length l is associated with every sampling point by concatenating its bit with the successive $l - 1$ bits, calculated with *sign* of temporal difference;

4. the LBP code in base 10 is computed.

$$LBP_c = \sum_{i=0}^{l-1} s(d_i)2^i \tag{1.6}$$

where

$$s(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & Otherwise \end{cases} \tag{1.7}$$

Fig. 1.6 shows the composition of a LBP code. Fig. 1.7A show 30 seconds of multichannels recording.

Fig. 1.7B show examples of LBP code with $l = 5$. Fig. 1.7C illustrates how histograms of LBP codes differ between interictal and ictal states. During the interictal state the LBP codes are well distributed over almost all the possible codes. In contrast the ictal window has a predominant portion of a single LBP code and many LBP codes are missing due to the typically slow and asymmetric oscillations evolving during seizures.

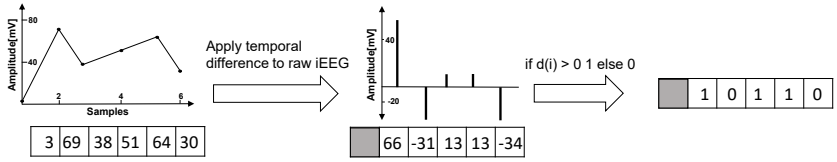

Figure 1.6: Starting from the raw iEEG signal I evaluate the temporal difference; later I construct the Local Binary Pattern code of 1 and 0, depending if differences is positive or negative.

Local Gradient Pattern Encoding

The Local Gradient Patterns are a different type of signal pattern encoding, that similary to the LBP one, try to encode the short-time and long-time dynamics of the signal. The LGP has been successfully applied in face detection [46] and, in the version that I am going do describe, in seizure detection [11]. The same considerations of LBP holds. The LGPs are computed as follows:

1. decide the number of points of the string of LGP (m);

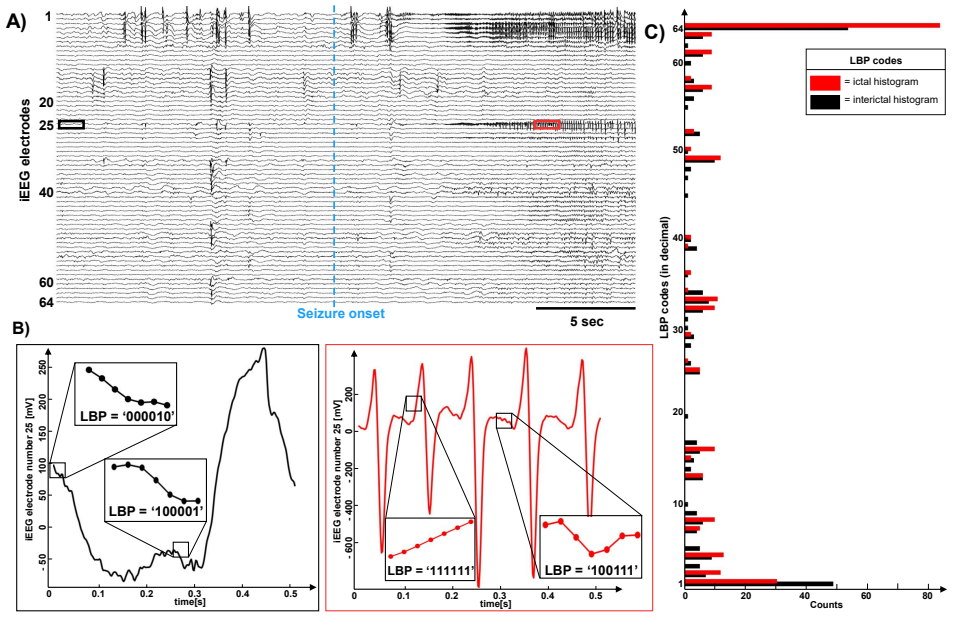

Figure 1.7: A) iEEG signals before (interictal) and after (ictal) seizure onset. The blue dotted line marks seizure onset as determined by the visual inspection of an expert (K.S.). **B)** Zoomed in iEEG signals: (1) during the interictal state: the LBP codes are well distributed over almost all the possible codes; (2) during the ictal state, the strongly time-irreversible signals have a predominant portion of a single LBP code; examples of their LBP codes of l=6 are drawn. **C)** The corresponding histograms of the LBP codes inside the 0.5 s windows in **B**.

2. for each point S_c, divide the m in $m/2$ forward points and $m/2$ backward points;

3. compute the gradient for each of that point as $g_i = |P_i - S_c|$

4. compute the mean gradient

$$g_{avg} = \frac{1}{m} \sum_{i=0}^{m-1} g_i \tag{1.8}$$

5. compute the gradient code as $gc_i = g_i - g_{avg}$.

6. compute the value in base 10 as for the LBP encoding.

Figure 1.8 shows the full procedure.

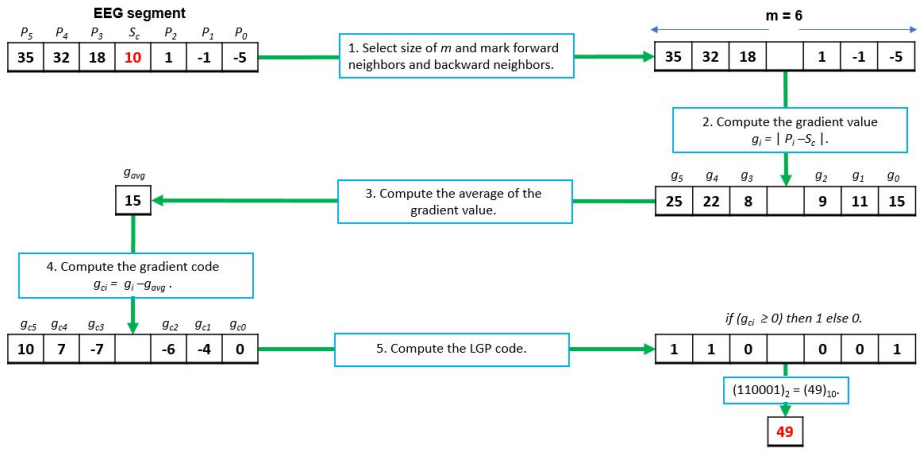

Figure 1.8: Shows all the step for the construction of LGP code for a single time sample S_c.

1.2.3 Short Time Fourier Transform

The short-time Fourier transform (STFT) [47], is a Fourier-related transform used to determine the frequency content of local sections of a signal as it changes over time. In practice, for computing the STFT you divide a whole time signal into shorter segments of equal length and then compute the Fourier transform separately on each segment. This give you the frequency components of the signal over time, in each single window.
In the *continuous-time domain*, the function to be transformed is multiplied by a filter window, which is nonzero for different segment every time and the Fourier transform is applied on this resulting function. The formula to

be applied is:

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)\omega(t-\tau)e^{-j\omega t}dt \tag{1.9}$$

where $\omega(t)$ is the time window and $x(t)$ the signal to be transformed.
In the *discrete time case*, the samples are divided in chunks and each chunk is Fourier transformed and added to a matrix, with time on raws and frequency on columns. This is expressed as:

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]\omega[n-m]e^{-j\omega n} \tag{1.10}$$

with $\omega(t)$ conitnuous and $x[n]$ the quantized and discretized signal.
In the figure 1.9 is reported the spectrogram of a single channel of a Patient: in the left part we can see the signal in the interictal stage, caracterized by low frequency components; in the righ part a seizure is shownwith increasing of the frequency higher components.

1.3 Spatial analysis of the brain

In a field where one third of the patient show seizure despite of optimal medical treatment, and where the surgery is the only alternative, the detection of the part of the brain from which the seizure start is crucial [5].
By now, using intracranial electroencephalography (iEEG), or any other diagnostic technique, the so-called *epileptogenic zone* (EZ), i.e. neuroanatomical areas that are necessary and sufficient to produce seizures, cannot be identified and completlely mapped.

One important challenge in this direction is to locate the seizure onset zone (SOZ, i.e the area where the the first EEG ictal changes are recorded), used in place of the EZ [48]. This could help the doctors in the decision of the resected brain tissue (RBT), depending both from the SOZ and from

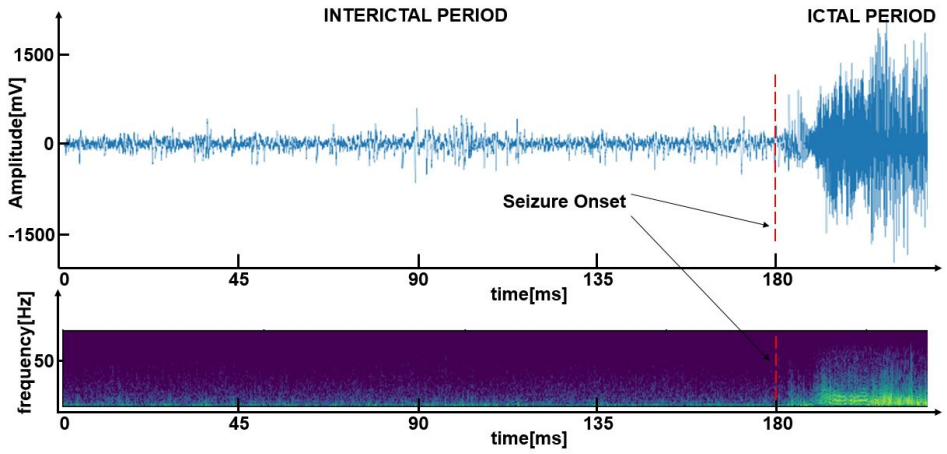

Figure 1.9: In the upper part of the figure the raw iEEG signal of 3 minutes of interictal period with the ictal transcation (Seizure Onset) is reported. In the lower part, the spectrogram of the short-time fourier trasform is computed and plotted. On the x-axis we have the time, on the y-axis the frequency component. A color much similar to yellow indicates an higher value of STFT. During Seizures we can see a clear increasing of the frequency of the signal.

the function of the tissue that they are going to remove, to avoid lost of abilities of the patient.
Looking to post-surgical follow-up, if the patient achieves long-term seizure freedom after epilepsy surgery, the clinicians assess that critical parts of SOZ/EZ have been included in the RBT.

To date, the clinical interpretation of iEEG recordings is based on expert visual analysis, with problems regarding time and expert-specific interpretation. Thus, an automated method to identify the SOZ could be of particular interest in this tricky field.

In my thesis I will propose an algorithm to identify the SOZ hemisphere of belonging and a possibility to go in a finer grain detection, with the identification of the SOZ.

1.3.1 Statistic

Here I will give the basic concepts of statistic applied to biological and life sciences [49].

At the end of this section, you should be able to understand all the test used and to interpretate their results: all these tests have been used to statistically identify the SOZ or the hemisphere of it. The input data are taken from an algorithm that we will describe after.

In the following, I give a brief introduction about the main concepts to understand the statistical tests:

- *Null hypothesis*: it is the hypothesis tested. The test could result in a statistically significant refuse of that hypothesis or not: it is important to underline that the acceptance of the null hypothesis could be caused by its truth or simply by the lack of data to demonstrate that is wrong.

- *Alternative hypothesis*: is the hypothesis that is confirmed by a refuse of the null hypothesis. It could be identified as the logical negation of the null hypothesis.

- ρ *value* is used to assess the significance of the test. The test will tell us to refuse the null hypothesis and use the alternative one if the ρ *value* is near to 0. Higher value of ρ *value* will indicate that the null hypothesis holds. By convention, the ρ *value* is set to 0.05 or 0.01 before performing the test to decide which level of confidence we have to accept: in this case, the experiment refuses the null hypothesis with a confidence level of 95%/99%.

Levene's test

The Levene's test is used to assess the difference of variances for a coefficient calculated for two or more groups. This test is necessary to verify some

assumptions done in other tests, like *t-test*, that could assume different or equal variances of the groups.

It test the null hypothesis that the population variances are equal. The result is expressed through the ρ *value*. Before performing the test you have to choose a value of it: for example, accepting with a ρ *value* < 0.05, involves a level of confidence of 95%. Values of ρ near to 0 implies to refuse the null hypothesis, i.e. the homogeneity of the variance. In other words, ρ *value* $\simeq 0$ implies that the two variance are different.

t-Student test

The *t*-student test gives us information about the mean of a population *normally* distributed.

There are mainly two types of *t*-test:

- **one-sample** *t*-test compares the mean of a sample population to an hypothetical constant mean; this test is used for *abnormal detection*. A researcher maybe interested in comparing the body temperature (37°) with the body temperature of people affected by pneumonia: it will than compare its samples with the mean body temperature of 37°. A ρ *value* < 0.01 tells the researcher that there is a statistically significant difference;

- **two-sample** *t*-test compares the means of two populations against each other to determine if they are different. Returning to the previous example, I want to assess if there is a different body temperature between healty patient and patient with pneumonia. The ρ *value* is used also here to assess the difference between the two means: the null hypothesis is mean of first group equals to the other mean.

The last important distinction in *t*-test is between one-tailed and two-tailed (figure 1.10), used respectively when you know which mean is the higher

and when you don-t know it. For my experiments in the spatial domain I use two-samples two-tailed t-test.

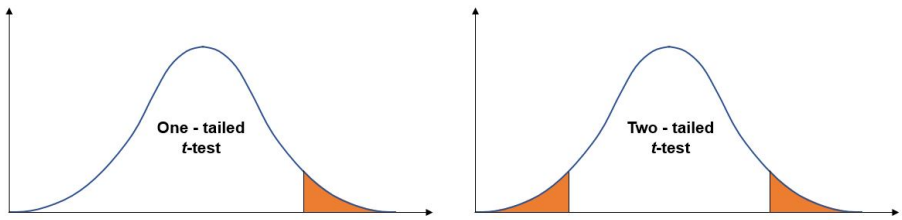

Figure 1.10: On the left, the one-tailed t-test: I take only the right part of the distribution (i.e. the t-student distribution): in this case we should be already familiar with the populations, knowing which one has higher mean. On the right, the two-tailed version, where we test the difference of the two means, without any knowledge a priori of them.

Mann-Whitney test

This test is made in alternative to the two-samples t-test, since it is the not parametric version. When the assumption of *normality* of the input groups doesn't hold this test has to be used to assess the difference between the means of the tow input groups.

This test needs only the independence of samples: if the distribution is normal, its efficiency is around 95%. All the considerations made for the t-test are identical.

One-way Anova test

One-way analysis of variance (one-way Anova) is used to compare means of two or more samples (using the F distribution). In particular, it is used in place of the t-student for the case of at least three groups. The One-way Anova makes the assumption that the groups are normally distributed and the variances of each group are equals.

The Anova tests the null hypothesis that samples in all groups are drawn from populations with the same mean values.

To do this, the Anova exploits different estimation of the variances: if the group means are drawn from populations with the same mean values, the variance between the group means should be lower than the variance of the samples, following the central limit theorem.

Nevertheless, the One-way Anova has again a ρ *value* as output: using it we can understand if all the groups have the same mean or not, but we can't infer on which are the groups with different means.

Hence, if the result of Anova refuses the null hypothesis, we need further tests to understand the groups with higher or lower means.

Kruskal-Wallis test

The Kruskal-Wallis test is a non-parametric method for testing whether samples originate from the same distribution. It is used for comparing two or more independent samples of equal or different sample sizes.

As the One-Way Anova test extends the t-test, it has the same role for the Mann-Whitney test. A *significant* Kruskal-Wallis test indicates that at least one population is stochastically different from the others. The test does not identify which is this group or how many distributions are used to drawn the input samples. For analyzing the specific population pairs you need further tests.

Since it is a non-parametric method, the Kruskal-Wallis test does not assume a normal distribution of the populations, unlike the analogous one-way analysis of variance.

The null hypothesis in that case, is that the medians of all groups are equal, and the alternative hypothesis is that at least one population median of one group is different from the population median of at least one other group.

Bonferroni test

The Bonferroni test is used to compare multiple groups in statistical analysis and is made usually after a One-Way Anova: if the Anova refuses the null-hypothesis we know that at least one group have different mean with respect to the others, but we don't know which one. The bonferroni test takes all the input group and perform pairs-comparison of the mean of each group: the output is a matrix, with all the ρ *values* for all the possible pairs. Thus, we can identify groups that are stochastically different from all the other groups.

Chapter 2

Algorithm

In this chapter, I will explain in details the whole architecture of my algorithm that exploits only binary operations to achieve very high accuracy with a low power consumption and a small memory footprint.

I use almost the same algorithm for both the datasets(Section 4.1) for seizure detection task (Section 2.1). For the SOZ identification (section 2.2), I integrate some statistical analysis to the codes extracted with the Seizure detection algorithm.

2.1 Seizure detection

The LBP feature extractor and HD computing are combined to quickly learn from ictal iEEG to then detect further seizures.

My proposed algorithm uses LBP codes to directly symbolize the iEEG signal of an electrode. Then a composite d-dimensional binary representation is constructed to capture the statistics of the LBP codes across all electrodes and over time. The final classification is followed by simple post-processing as shown in Fig. 2.1.

Overall, I am going to present a method for efficient seizure detection, with end-to-end binary operations:

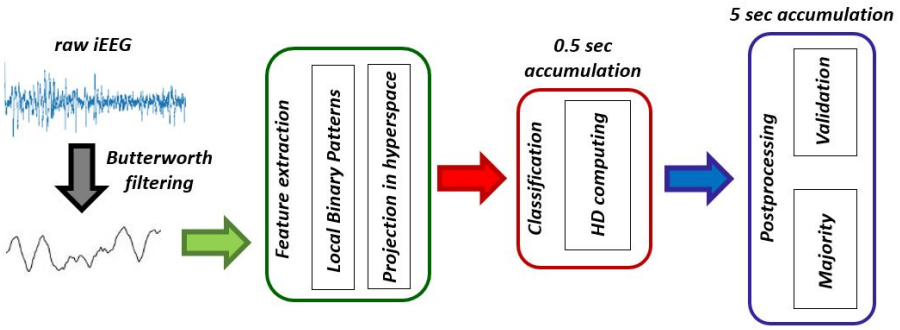

Figure 2.1: Binarized seizure processing chain: (1) Feature extraction generates a 6-bit LBP code for each electrode and projects these codes into d-dimensional space; (2) HD processing constructs vector H, which represents the histogram of 0.5 s recording; (3) As postprocessing, a simple patient-dependent majority voting decides based on the last 10 labels; for the long time recordings a validation of that window as been added to reduce the false alarms.

1. the *Feature extractor* immediately transforms the iEEG signal in binary symbols, avoiding the high power consumption that could derive by computing with floating point arithmetic. There are others very popular feature extractor in this field, like wavelet transformations [50–52] or stft [13], but they are all based on floating point operations.

2. the *classifier*, again, is very simple and built with binary operations: it projects the LBP in the hyperspace (i.e. binary space in $10k$ dimensions) and perform classification using something that is very similar to the the nearest mean, using a specific metric in that hyperspace.

3. the *Postprocessing* is composed by two different parts: a simple majority on a big time-window and a *validation* of that window. I will explain in details in the section 2.1.3.

The timing of the algorithm is very simple, and all the operations are executed in parallel fashion (see figure 2.2):

- a window of 6 samples is used to construct the LBP for every time sample: the overlap is maximal, sample by sample;
- all LBP for a window of 0.5 seconds (1 second for the long time algorithm) are accumulated and used by the classifier. This window is sliding with no overlap for the short recordings and with a 50% overlap for the long time ones.
- a final window of 5 seconds is used for Postprocessing, with maximal overlap with respect to the output of the classifier, i.e. sliding of 0.5 seconds at the time.

2.1.1 Preprocessing and LBP Feature Extraction

The first step is to extract the electrical activity of the brain: in the section 4.1 we will see in details how electrodes are used for this task and how the process of extraction works.
For my purpose, the iEEG signals are converted from the analog domain by a 16-bit ADC, filtered by a fourth order Butterworth filter between 0.5 and 150 Hz, and downsampled to 512 Hz.
All the recordings on which I work are multi-channel and constructed with this procedure.
A LBP code with $l = 6$ is computed for every sampling point. The LBP code considers six consecutive samples, and moves by one sample. My LBP code generates 2^l different symbols that are fed into the next stage for learning and classification: it is important to notice that a lot of that codes are never generated and the total *size* of the alphabet is less than 2^l. Using larger code sizes impairs its applicability to non-stationary signals and latency of classification. Using smaller ones reduce too much the number of

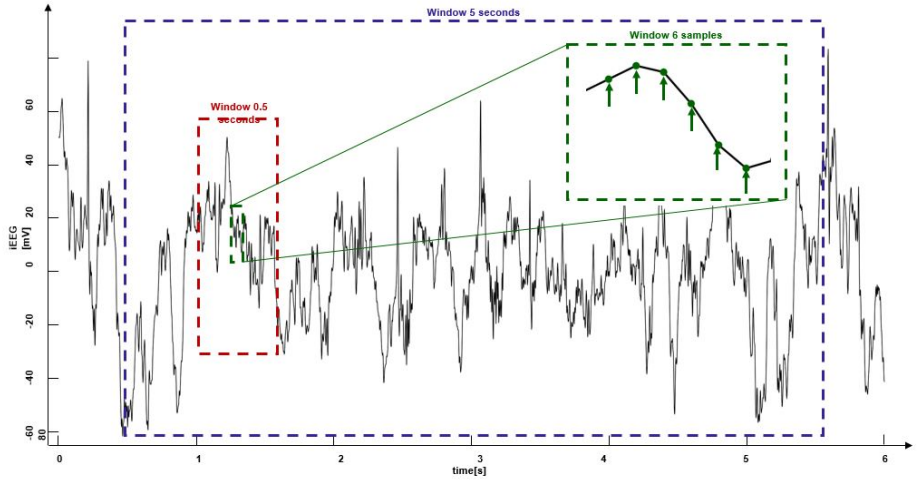

Figure 2.2: iEEG signal of a single channel. In green the window of points used to construct a single LBP symbol, shifted of 1 sample at a time (256 symbols inside a window of 0.5 seconds); in red the 0.5 seconds not-overlapping window used to construct the histogram; in blue the bigger time-window (5 seconds length, 0.5 seconds shifting) used to compute the final prevision.

extracted *features*.

The code size determines also the minimum size of following statistical analysis window: the size of such window should be large enough such that all symbols can at least theoretically occur once [23]. Hence an analysis and classification window of 0.5 s (containing 256 samples) is sufficient ($256 > 2^6$). So, the function of the first block is to convert an input scalar value to a string of 6 bits, i.e. the LBP code. After that, HD computing first projects the LBP codes to the high-dimensional space via an Item Memory (IM_{LBP}) : the IM is a matrix of dimension $64 \times d$, where d is the dimension of the hyperspace composed by binary orthogonal vectors, one for every to every LBP code, i.e., $C_1 \perp C_2 \ldots \perp C_{64}$. The matrix is directly indexed by the LBP code: the results is the projection of each symbol in

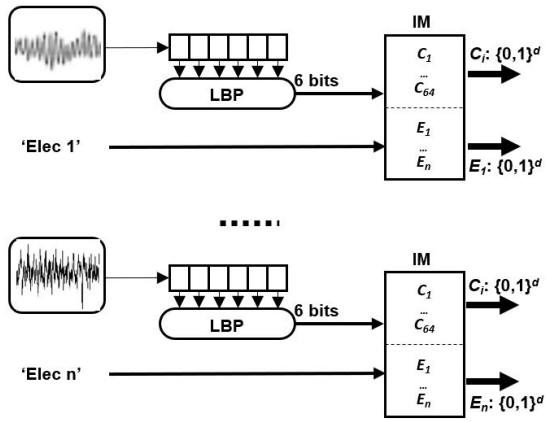

Figure 2.3: In this stage I extract from each electrode the LBP code and the electrode name (i.e. the ID). After that, I project them in the hyperspace through two item memories properly constructed.

an $10k$-dimensional vector, suitable for the next step of the architecture.
To combine these vectors across all electrodes, HD computing generates a spatial record (S), in which an electrode *name* is treated as a field, and its LBP code is treated as the value of this field.
Another IM_{ELEC} is used to map the name of electrodes to orthogonal vectors: $E_1 \perp E_2 \ldots \perp E_n$ for a patient with n electrodes. In this case the dimension of the item memory is linearly dependent from the number of channel of the patient.
We can conclude that this stage of the algorithm transforms n scalar values in n value-hypervectors,that represent the scalar points inside their sequences and n key-hypervectors, that are used to encode the name of the different channels.

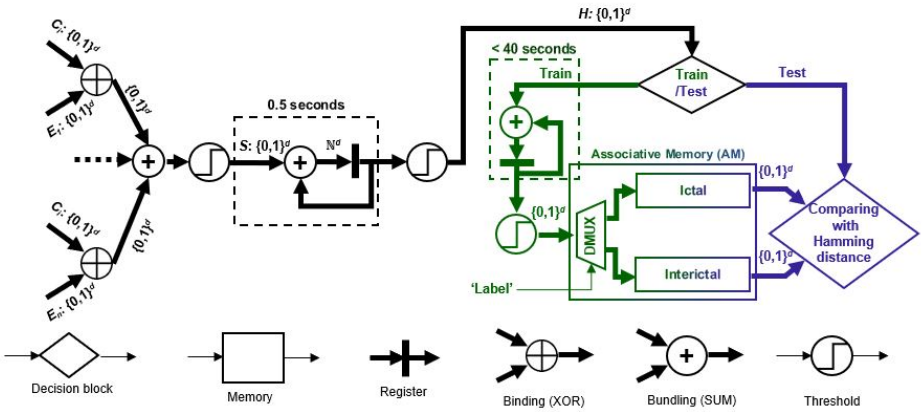

Figure 2.4: Starting from the hypervectors of name and values of electrodes, the classifier construct S: this hypervector encodes the spatial information among all the electrodes. After that, all the S in a window of 0.5 sec are added and the *histogram* H is created. Finally, there is a split: during the training phase all this H will contribute to construct the AM; conversely, during the test, the classifier gives an initial label to the window.

2.1.2 HD Learning and Classification

In this second stage I work on the output of the previous block to produce an initial classification of the iEEG.
In the HD space there is the possibility to exploit the multiplication (section 1.1.2) to store data and keys in the same location.

Example 3 *Takes* $A = 10$ *and* $B = 20$. *I associate two hypervectors to the two values* $V1, V2$ *and two to the names* A, B. *In the traditional way of computing, the two values are stored in different locations that could be indexed by the name of the variables. In the HD paradigm, we store all in a single hypervector* $S = A \times V1 + B \times V2$ *and we recover the two values by multiplication with* A *and* B.

The example 3 allows to bind the name of each electrode ($E_j \mid j \in [1, n]$)

to its corresponding code ($C_i \mid i \in [1,64]$).
The spatial record (S) is then constructed by bundling these bound vectors using majority gates: $S = [E_1 \oplus C_i + E_2 \oplus C_i + ... + E_n \oplus C_i]$. In other words, I obtain n bound vectors and I add them together summing and comparing componentwise the n bits through a majority gate: hence, there are d majority gates working on n inputs.
Vector S is computed for every new sample, and represents the spatial information about the LBP codes of all electrodes. S takes place of the vector of features in the hyperspace: we have a single hypervector of binary values instead of a vector of scalars representing the features.

The next step is to compute the histogram of LBP codes for a moving window of 0.5 s (1 s for long time recording). This window size should be large enough to theoretically permit at least a single occurrence of all possible LBP codes [23, 24]. The window of 0.5 s contains 256 LBP codes constructed with maximal overlap. To have a high probability that every code occurs inside this window, we should hold $256 > 2^{l+1}$, hence $l < 7$.
To estimate the histogram of LBP codes inside this window, a multiset of temporally generated S vectors is computed as $H = [S^1 + S^2 + ... + S^{256}]$. A majority gate is applied in the temporal domain through accumulation (i.e., componentwise addition) of S^t vectors $t \in \{1, ...,256\}$, that are produced within the window, and then thresholding at half.

We observe that the interictal and ictal states show different distributions of LBP codes inside the window: during a interictal segment, we have a nearly random signal, with a well distributed count histogram; conversely, during a seizure we typically observe rhythmic signals, i.e., slow and often temporally asymmetric oscillations, which yield polarized histograms as demonstrated in Fig. 1.7 of section 1.2.2.

This shows that the distribution of LBP codes, not necessarily their sequence, is an important indicator to distinguish ictal vs. interictal states. So I can get rid of the distribution in time of the S, simply adding them. From a theoretically point of view, the histograms features of the LBP give a good classification because intrinsically encode two properties of the seizures:

1. **time irreversability** [42]: as said when talking about dHVG, this behavior is typical of ictal periods. A high quantity of symbols $'111111'$ with respect to $'000000'$ encodes perfectly this feature: in fact, since the first LBP code encode the monotonic increasing of the signal and the second the monotonic decreasing, the high imbalance of the two means that the signal is not symetric for which it concern the time;

2. **entropy** [23]: I introduce here this new concept, the permutational entropy of the signal, expressed as

$$H = -\sum_{i}^{N} p_i \times log_2 p_i \tag{2.1}$$

   where i goes over all the possible local binary pattern codes and p_i is the probability that the target code appears in a target time window. A variation of thie entropy, mainly due to more rithmic signals, has been demostrated to be correlated with the seizure events: of course storing the whole histogram will be more than enough to catch this variation in the entropy.

Back to the HD computing, the high-dimensional space can naturally encodes such histograms in H by accumulating and thresholding the spatial vectors. To assess the similarity of real histogram with H, I compare the last one with all symbols inside IM_{LBP} and calculate the normalized Hamming distance as relative frequency of the symbol. The reconstructed histograms

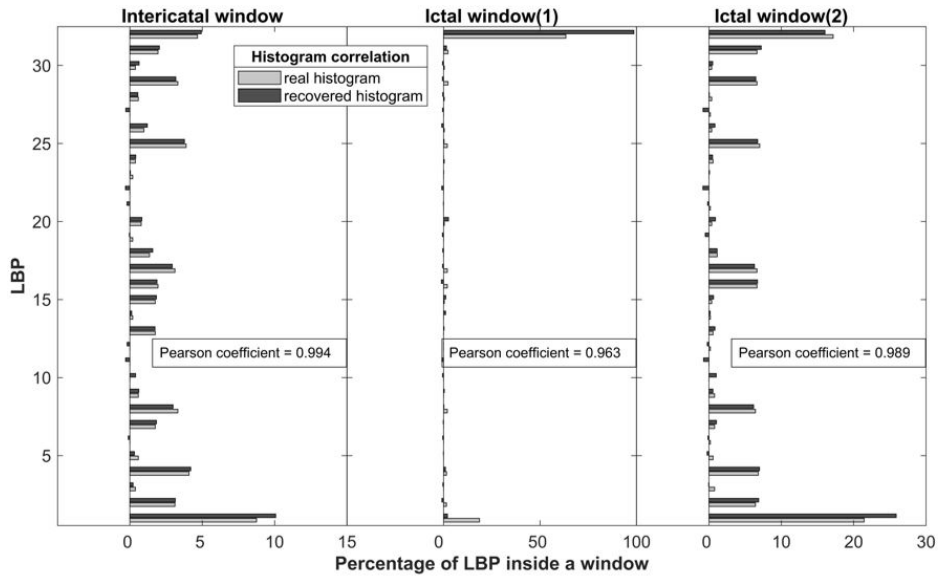

Figure 2.5: Three windows are first encoded using LBP ($l = 5$) and then recalled through HD; all the correlations, in interictal and ictal periods are so high.

achieve a Pearson correlation coefficient > 0.9 compared to the exact histograms. Some examples from single channels during ictal and interictal periods are shown in figure 2.5

The output of HD encoding is vector H, which is updated every 0.5 s.

Now, the classifier is divided, depending on the function that is performing:

1. during the training I use the H vector to build the Associative Memory containing two *prototype* vectors representing ictal and interictal labels.

   To train the interictal prototype, all H vectors computed over an interictal window of 40 s are accumulated (summed), and then thresholded (binarized) to be stored in the AM. Correspondingly an ictal prototype vector is generated from a smaller window of 10–30 s depending

on seizure duration. If more than one seizure is used to train, I generate a prototype for each seizure and then I compose them together by addition, having again a single vector to store in the AM.

2. For classification, a newly computed H vector is compared to every prototype of the AM using normalized Hamming distance to determine its label. Exactly as in the nearest mean algorithm, the nearer prototype to the query H gives its label to it.
   Also here, it has to be highlighted the simplicity of classification: rather than using multiple floating point operations as done in deep learning, a componentwise $\oplus$ and a scalar comparison are enough.

2.1.3 Postprocessing

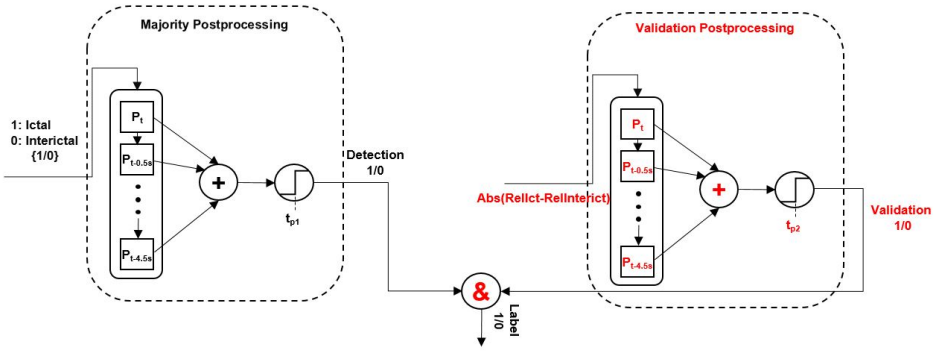

Figure 2.6: Both windows of postprocessing of the algorithm: the left one, a majority on the last ten samples, is used for both the datasets; the validation postprocessing has been added to avoid a high number of false detections.

The last part of the algorithm postprocesses the labels produced by the HD classifier every 0.5 s.
The goal is to reduce the number of false detection during the interictal period: in fact, rising too many alarms could increase too much the level

of anxiety of the patient and the negative effects would be higher that the benefit that a device implementing this algorithm could offer.
It defines a window of 5 s (shifting of 0.5 s at a time) where a final decision is made based on the last 10 labels collected from the HD classifier. Nevertheless, the *reliability* of these labels is used to validate the window of prediction. Let's distinguish this two type of postprocess:

- the first one, the only used for the short recording dataset, is a simple majority window: seizure onset is detected only if the number of ictal labels in the last 10 samples is equal or higher than a predefined patient-specific threshold (t_{p1}) where $t_{p1} \in [8,10]$ (figure 2.6, left).
  This window is used to discard all false seizures that could randomly arise from the very noisy EEG signal.

- the second, identified as *Validation postprocess* (figure 2.6, right) has been added to further reduce the number of false alarm in long time recordings: taking the similarity between the query and both the prototypes, I can calculate the reliability as

$$\text{Reliability} = |\Delta(H, AM1) - \Delta(H, AM2)|. \tag{2.2}$$

  The window is validated (i.e. could be predicted as ictal) only if this value is higher than a second patient-specific threshold (t_{p2}).
  Forcing this validation means to predict a seizure only if I am so sure that is happening, i.e. the H is very similar to the correct AM prototype.

The window size chosen provides a trade off between the delay of detection and false alarms.

Overall, my algorithm has four parameters: the size of LBP code, the duration of the two windows, d, and the 2 threshold t_{p1} and t_{p2} . Only the last parameter, the thresholds, is patient-dependent, whereas the others are

fixed for all patients. Nevertheless, to reduce the memory load d can be adjusted to the individual patient depending on the number of electrodes and seizure dynamics. In the chapter Results I will tune this dimension d to the minimum patient-specific, with whom the performances are maintained.

I observe that the algorithm works with $d = 10{,}000$ for all patients. For some patients it may even be reduced to 1000 without impairing its performance, but with a big gain in power and memory footprint.

2.1.4 Patient specific application: spatial Item memory

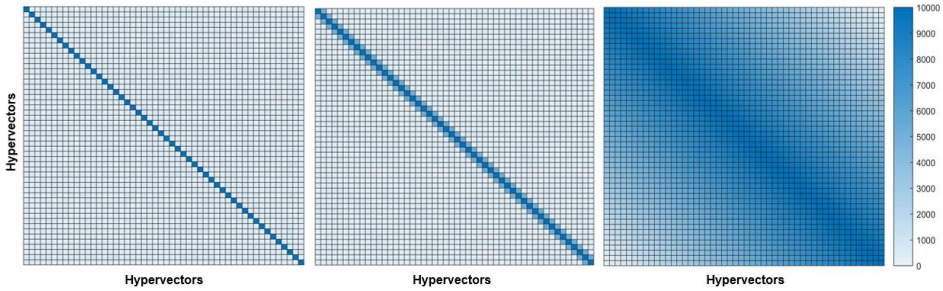

Figure 2.7: Represent the cosine similarity (or normalized hamming distance, for binary vectors) computed with vectors inside different item memories. From left to right: **Random item memory**, with all the hypervectors orthogonal to each other, **Sandwich item memory**, where each hypervector is similar to the two neighbors, the one before and the one after, but orthogonal to all the others, and **Continuous item memory**, that well represent a range of values: each hypervectors is more similar to the immediately neighbors and less to more distant ones.

The algorithm described in the previous sections is very general: no assumption has been made on the position of the electrodes: each element has been assumed to be equally distant to each other.
However, this assumption doesn't reflect reality: when the electrodes are implanted in the brain, you know which is more near or more distant. Furthermore, these implanted electrodes are divided in strips/grids and you

can keep trace of this information.
This fact could impact the performances because successive seizures in time could be recorded by very near channels: if I keep trace of the position of the electrodes, I will detect them to be very similar.

Example 4 *Let's assume that you have a prototype, composed of 2 values (i.e. two orthogonal hypervector) linked to the respective channels, i.e.* $prototype = A \times V1 + B \times V2$ *and a query,* $query = A \times V3 + B \times V1$*, and you want to compare them with cosine similarity. The new vector will be*

$$\begin{aligned} S &= prototype \times query \\ &= V1 \times V3 + V2 \times V1 + A \times B \times V2 \times V3 + A \times B \\ &= noise + noise + noise + A \times B \end{aligned} \tag{2.3}$$

where noise points to new random and orthogonal hypervectors. It is now easy to see that if A and B are ortogonal, I totally lose the fact that V1 was in two near channels; conversely, if A and B have some similarity, I can mantain this information. In the former, S would be a vector with equal number of 1 *and* 0*. In the latter, S has an higher quantity of* 1*.*

It is obvious from example 4 that keeping trace of position of channels is fundamental to increase the performances of a patientspecific algorithm.
Let's understand how this can be exploited in the HD computing, analyzing different type of IM (figure 2.7):

- **random item memory**: this is the one currently implemented; all hypervectors are orthogonal with each other and no information is assumed on the spatial proximity;
- **sandwich item memory**: this type of IM models the hypervectors before and the one after as nearer to the target hypervectors and all the others at the same distance. This configuration can catch all the equal signals showed on the 2 neighbor channels: if the prototype has

a target signal on channel 4 and the query has it on channl 3 or 5, this memory will catch it;

- **continuous item memory**: is the last 'straightforward' solution; the similarity of two hypervectors is directly proportional to their distance inside the IM. This relation well encode a set of items equally distributed on a straight line.

Starting from this initial configurations, I propose a possible solution to create a patient specific item memory: I extend the concept of sandwich memory, making bigger zone of similarity based on the continuous IM.
In figure 2.8 we can see an example: this memory encode an artificial situation, where we have 50 electrodes in the brain composed by five 6-electrode strips, one 8-electrode strip and one composed by 12 electrodes (a strip is a set of electrodes disposed in a straight line).
In a possible future implementation on a patient, this adjustment could significantly increase the sensitivity of the algorithm.

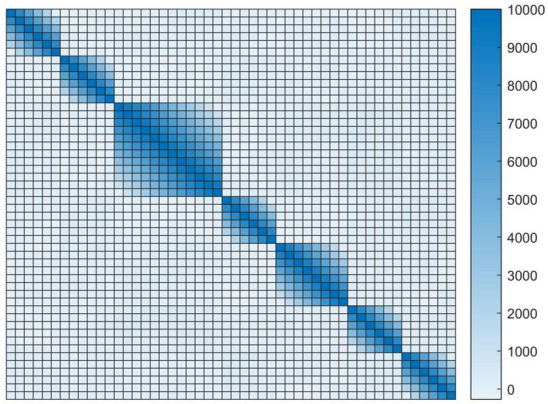

Figure 2.8: IM designed to encode the similarity between electrodes in five 6-electrodes strips, one 8-electrods strip and a 12-electrode one.

2.2 Seizure Onset Zone identification

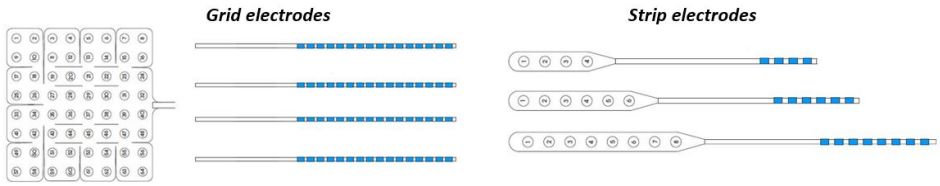

Figure 2.9: Grid and strip electrodes. After surgical implantation, the electrical activity is recorded through the blu pins on wires that comes out from the heads. Patient with this type of electrodes are under constant monitoring.

In this section, I present the main contribution of the thesis for identification of ictogenic brain regions. In a field where one third of the patient show seizure despite of optimal medical treatment, and where the surgery is the only alternative, the detection of the part of the brain from which the seizure start is crucial [5]. Precise identification of ictogenic brain regions followed by surgical resection often improves seizure control and can even eliminate the occurrence of seizures completely [5].

By now, using intracranial electroencephalography (iEEG), or any other diagnostic technique, the so-called *epileptogenic zone* (EZ), i.e. neuroanatomical areas that are necessary and sufficient to produce seizures, cannot be identified and completlely mapped. An important practical challenge is that with presurgical iEEG recordings (or any other current diagnostic method) the brain tissue of the so-called “epileptogenic zone”, i.e. neuroanatomical areas that are necessary and sufficient to generate epileptic seizures, cannot be mapped directly and completely. Therefore, in clinical practice, the seizure onset zone/"ictogenic zone" (SOZ, i.e. the area where the first ictal iEEG signal changes are recorded), is used as a proxy for the epileptggenic zone [53].

To date, this practice is done by the clinical interpretation of iEEG

recordings, mostly based on visual analysis, that is time-consuming and may yield high variability: using the iEEG coming from the implantation of grid and strip (figure 2.9) electrodes, they record the electrical activity over the brain of the patient and select the channels (electrodes) which show an emphatic ictal behavior.

This demands an algorithm that can learn *transparent* codes from the iEEG recordings, hence the codes are analyzable to locate the SOZ. HD computing produces such codes with interpretable features due to its well-defined set of arithmetic operations with inverses. In the following, I describe my algorithm that can automatically identify the SOZ at two levels of spatial resolution: the cerebral hemispheres and lobes.

2.2.1 Hemishpere identification

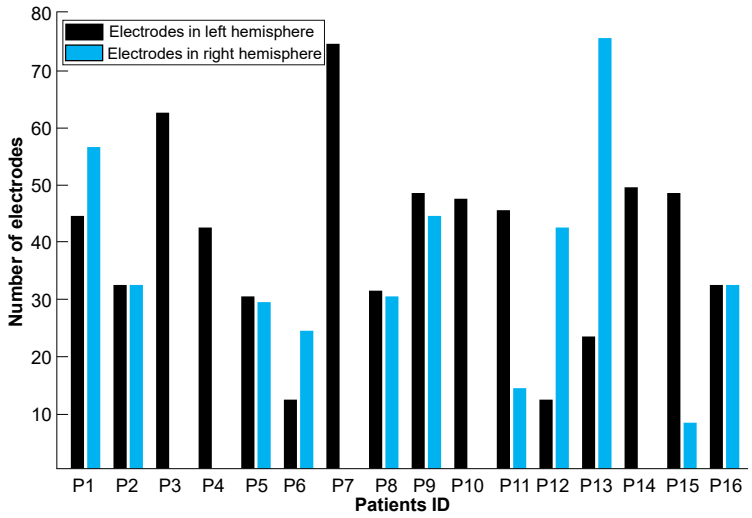

Figure 2.10: Number of electrodes implanted in left (black) and right cerebral hemisphere (blue).

Fig. 2.10 shows the number of electrodes implanted in the left or right

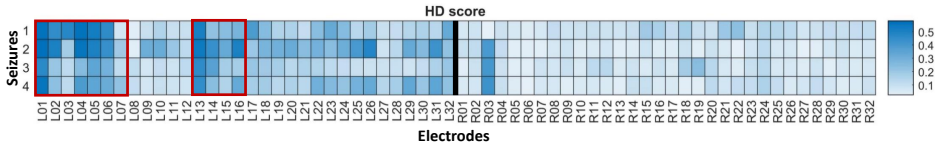

Figure 2.11: Electrodes' scores of P2, generated by four subsequent seizures. Considering only the first seizure (the first row of scores) is sufficient for the algorithm to identify the correct hemisphere. The red squares mark the electrodes in the SOZ. The box-plot of scores is shown in Fig. 2.12.

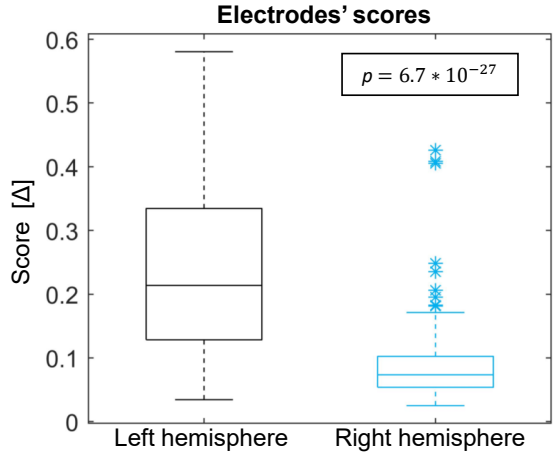

Figure 2.12: Box plot with scores of P2 by supplying all four seizures. On each box, the central mark indicates the median, and the bottom/top edges of the box indicate the 25th/75th percentiles. The dotted line extends to the most extreme data points not considering outliers. Outliers are marked with '*'.

cerebral hemispheres. As shown, for 11 patients out of 16, the clinical experts have no initial are uncertain whether the seizures start from the left, from the fight or from both cerebral hemispheres and hence the number of electrodes implanted into both hemispheres is almost equal. My first aim here is to identify the location of SOZ at the spatial scale of hemispheres.

My proposed algorithm is mainly based on the seizure onset detection algorithm (Section 2.1) followed by statistical hypothesis testing. The algorithm first generates two sets of ictal (P_j^S) and interictal (P_j^I) prototypes for every electrode j by bundling the vectors representing the LBP symbols ($C_{w,t}^i$) extracted from the related segments:

$$P_j^S = [\sum_{w \in ictal} [\sum_{t \in window} C_{w,t}^i]] \mid i \in [1{,}64]\ j \in [1, n]$$

$$P_j^I = [\sum_{w \in interictal} [\sum_{t \in window} C_{w,t}^i]] \mid i \in [1{,}64]\ j \in [1, n]$$

Note that the prototypes are computed solely for the relevant electrode. For example, for the 'electrode 1', one prototype vector (P_1^I) is associated with the interictal segment, and another prototype with the ictal segment (P_1^S); if the algorithm requires more ictal examples multiple ictal prototypes will be generated (one per ictal example). As is immediately apparent, when two different electrodes k and l continuously receive the same input stimuli, their corresponding encoded prototypes are identical, i.e., $P_k^I = P_l^I$ and $P_k^S = P_l^S$.

Then, the algorithm computes at least one score doe every electrode j as the normalized Hamming distance between its related prototypes ($\Delta(P_j^I, P_j^S)$): the larger the identification score, the closer the proximity of the electrode to the SOZ. When the two prototypes encode iEEG segments with very similar dynamics (i.e. the same distribution of C^i), they yield a $\Delta \approx 0$. Conversely, a $\Delta \approx 0.5$ implies no correlation between the two prototypes due to a different distribution of the C^i. In other words, a $\Delta \approx 0$ indicates that the electrodes are not involved in the seizure activity whereas an increasing value of the score suggests an increasing proximity of the electrodes to the SOZ. Fig. 2.11 shows an example of the scores computed for each electrode from four seizures of P2.

Finally, the algorithm incorporates statistical analysis with the computed scores of individual electrodes for identifying the seizure-generating hemisphere. The electrodes are divided into two groups, i.e. electrodes in the right hemisphere and electrodes in the left hemisphere. Subsequently, I use the Kolmogorov-Smirnov test to assess the distribution of the scores. Since the distribution is normal, I perform a t-Student test to determine if the two groups of scores are significantly different from each other. If the result of the test is statistically *significant*, i.e., the resulting p-value of the test is lower than 0.01, I accept it, and identify the hemisphere with highest score as the ictogenic one. Fig. 2.12 shows an example of the box-plot of the scores for P2, clearly indicating that the left hemisphere is ictogenic with p=6.7e-27.

If the result of test is not significant ($p \geq 0.01$), I reject it; this means that the algorithm requires more ictal segments to be able to infer the hemisphere correctly, hence I supply more seizure examples. For each ictal segment a separate prototype will be generated that effectively increases the number of scores per electrode and therefore the significance of the test. Therefore, my algorithm keeps computing the ictal prototypes and related scores over time until $p < 0.01$. For most of the patients a single ictal segment is sufficient (see Table 4.8).

Let's briefly summarize the full procedure: (1) I created the scores in the aforementioned way; (2) I visually define the normality of the data and I use the Levene's test to see if the variance of the data for the two score is equal; (3) using a t-student test with $\alpha = 0.01$ I compare the two means of the right hemisphere electrode scores and of the left hemisphere electrode scores to assess if one of them have a score that is reliable higher with respect to the other.

2.2.2 Channels identification

Here, my aim is to further investigate the location of SOZ at the spatial resolution of cerebral lobes. I straightforwardly extend my proposed algorithm (Section 2.2.1) by dividing the electrodes into a larger number of groups (hence using a finer spatial resolution), and accordingly invoke a proper statistical testing method. Since the generation of prototypes and computation of scores are the same as in the previous section, here I present the statistical analysis only.

First, the electrodes are divided into different groups according to the lobes that they cover. However, the exact electrode's membership in the lobes is not available, hence I use strips/grids (series of close electrodes) as a proxy. The strips with 6 or less electrodes are associated to a single group whereas the strips with 8 or more electrodes are divided into two different groups. Then, one-way ANOVA test (with Bonferroni post-hoc test) is performed to determine if at least one group of scores (i.e. one lobe) is significantly different from the others.

I use all the available ictal segments to compute the scores for this fine-grained identification.

Chapter 3

State of the art

In this chapter, I introduce to the reader the main competitors in this field: till now, there are many algorithms in the litreature, but almost any of them have been validated on a large dataset to assess their real performance in detecting seizures without rising continuous false alarms.
The majority of the seizure detection techniques [11,12,54,55] achieves very high results (near to 100% accuracy): however, all of them are tested on the very small dataset described in [22].
Using this dataset, they are far from the real life application, mainly for two reasons:

1. they all use a kcross validation technique, using a lot of training data with respect to testing;

2. the number of ictal segments and interictal ones are equal. However one challenge in this field, as for other classification tasks, is the imbalance of the classes, the presence of more istances of one class with respect to the other [56].

Nevertheless, the importance to learn from a reduced dataset is important, since the number of *relevant* events is low.

In the following, I focus on recent machine-learning based seizure onset detection methods [11–13] that may be partitioned into two groups. The first group is based on extracting time or frequency features or a mixture of them from the iEEG signals followed by traditional supervised machine learning methods such as support vector machines (Section 3.1) and artificial neural networks (Section 3.2). For implementation of this group, I use Python 2.7 with the Scikit-learn library. The second group consists of recent deep learning algorithms that automatically extract features and construct classifiers (Section 3.3 and Section 3.4). The deep neural networks are implemented in Python 2.7 using Keras library with Tensorflow backend. I added a final postprocessing step to all the competitors to avoid a too high number of false alarm in a long term scenario.
The postprocessing stage is also present in the majority of the works on dataset with long times recordings for seizure detection or prediction [13, 50, 57, 58].

3.1 Support Vector Machine with Local Binary Pattern

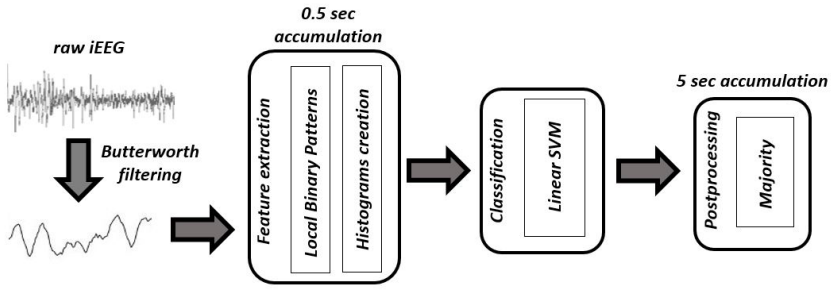

Figure 3.1: Pipeline for the SVM: the steps are identical to the ones of HD computing, only with a substitution of the classification paradigm.

The SVM is one of the simpliest classifier: it is a discriminative algorithm of machine learning, that create margins between classes: in particular, the SVM try to maximize these margins through orthogonal projection. In other words, the SVM creates an hyperplane in $\mathbb{R}^n$, where$n$ is the number of features, to separate the two classes.
The number of parameters, i.e. the coefficients of the hyperplane, is equal to the number of input features: each of them is trained through the gradient descendent, a technique that minimize the output error of the classifier. The formula that I want to minimize is:

$$[\frac{1}{n}\sum_{i=1}^{n} max(0,1 - y_i(w \times x_i - b))] + \lambda||w||^2 \tag{3.1}$$

where n is the number of samples, y_i the real output of sample i, w the weights, X_i the features of sample i and λ a regularization factor. The last important point to highlight is the dot product $w \times x_i$; the SVM could be caracterized by a so called 'kernel': changing it, I change the type of margins that the SVM draws.
Using a *linear* kernel, the dot product is mantained: conversely, changing the type of kernel, the $\times$ is substituted by the relation of that particular kernel.
The model has been trained with the linear kernel, using the box constraint C equal to 1.0 [11]. More complex kernels, as the RBF one, will result in an higher overfit due to the strong imbalance of data.
The SVM receive as inputs $n * Ch$ features, where n is the number of different LBP codes and Ch the number of channels. Each feature is drawn by a counter that keeps trace of the frequency of a single LBP code for an individual channel. In other words, the SVM works on the concatenated histograms of all the channels.

Working in this direction, I am directly comparing the Support Vector Classifier with the Hyperdimensional Computing, since both pre- and post- processing are the same; in this way, I can compare two hardware friendly techniques, demostrating that HD can outperform SVM for this type of classification.

3.2 Multi Layer Perceptron with Local Gradient Pattern

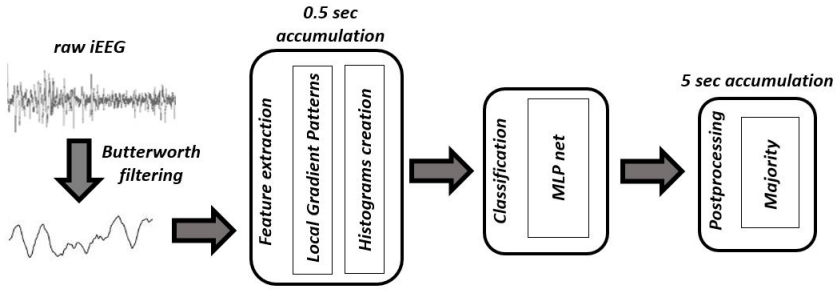

Figure 3.2: Pipeline for the MLP: the steps are again similar to the ones of figure 2.1, with a changing in the symbolization for the preprocessing, changing the local binary patterns in local gradient patterns.

The SVM divide the space of the solutions with a linear margin: the next, natural step, is to try to use convex margins to divide the space, before going to a total non-linear separation.
A feed-forward neural network does exactly this job if composed by a single hidden layer.
A MLP net (i.e. a feed-forward net) is composed by a connected net of neurons (figure 3.3) with a regular structure:

- a trainable weight w for each input of the neuron;

- a bias factor b;

- an activation function f that receive as inputs $\sum_{i=0}^{n} w_i \times x_i - b$ and trasform it to avoid too high numbers in the output.

Hence, my multilayer perceptron neural network is composed by three layers. One input layer, one hidden layer, composed by 40 neurons, and the output layer. The number of nodes in the input layer is equal to the dimensionality of the features space.
The training of the net is slightly more complicated with respect to the svm. The MLP exploit a generalization of the gradient descent, the backpropagation of the errors.
From a practical point of view, I feed the training points in my net in epochs: at each epoch, I calculate the output for each point and then I correct the weights with the reported formula:

$$w_{i,j}^{I} = w_{i,j} + \eta \delta_j \frac{df_j(e)}{de} x_i \tag{3.2}$$

where w represent the weight, η the learning rate, δ the backpropagated error and f the activation function of the neuron.
The training is stopped by some criteria that could be set, like a maximum number of iterations or the performance of the net on a cross-validation set.
As the SVM, also the MLP net needs an additional phase of preprocessing. As reported in [11] I use the LGP encoding, a different version of Binary pattern, that gives slightly better performances with this classifier.
Later, the features are constructed in the aforementioned way, feeding the MLP with histograms of channels. The simple pipeline is presented in figure 3.2.

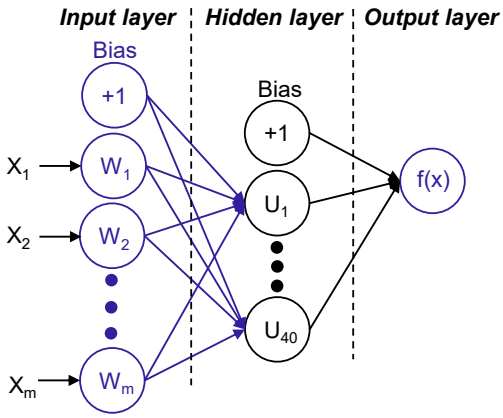

Figure 3.3: Multi Layer perceptron net is a fully connected net: the input layer is composed by m nodes, where $m = \#(channels) * \#(LBP_codes)$, the hidden layer is composed by 40 neurons, the output estimate the probability of the input to belong to the ictal class.

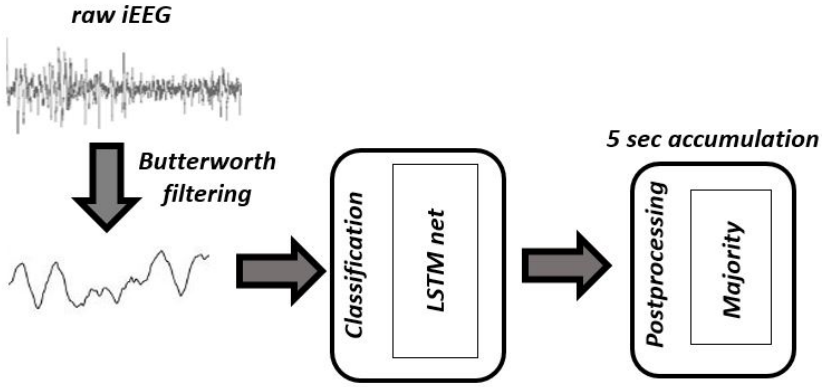

Figure 3.4: The pipeline based on the long-short term memory pipeline is simplier: the net directly works on the raw signal, extracting automatically the features from the signal.

3.3 Long-short term memory

As the previous two techniques, the RNN is taken from literature [12] on seizure detection, previously tested on the Dataset published by the Bonn University and adapted to my datasets.
This technique explot a unit, the LSTM, to create a network that keeps into account the time correlation between points.
Without entering in details, I list only the fundamental part of the LSTM neurons to understand its operation:

- an input gate receives the raw inputs and multiply them by a trainable matrix;
- a recurrent gate receives output of a previous neurons and multly again them with one other trainable matrix;
- an output output miesx the informations from the current trained time sample and the sequence informations coming from the recurrent gate to encode the whole new sequence on the output.

From this list, it is easy to understand that the LSTM is not a parallell model, but a sequential one: incremeanting the size of the window analyzed, the necessary encoding time increases.
Nevertheless, this model could automatically extract good features related to the timing correlations of ictal and interictal sequence, as shown from the good classification results.
As already said, the base version of the long-short term memory RNN works on single channel recordings, taking big windows of 23.6 seconds of raw iEEG signal [12].
This basic block has been mantained, to exactly compare my proposal algorithm with this method.
However, some modifications, in order to adapt the net to a real time detection with a multi-channel recording system has been applied: to be totally

fair, I try two different configurations, collecting results for both of them.

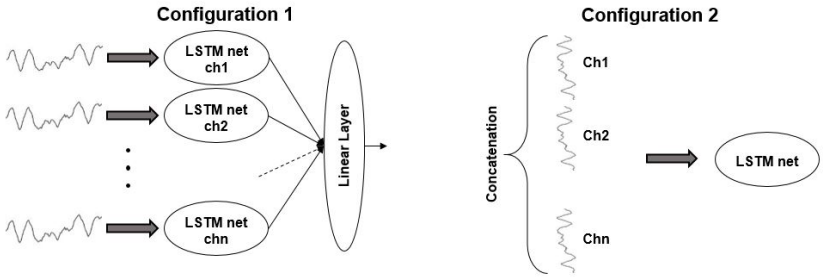

Figure 3.5: Long-short term memory configurations. In the first configuration, a single LSTM is used for each channel and the output are then merged through a Linear Layer. In the second configuration, the LSTM works on a bigger time window, created with the concatenation of all the input channels.

3.3.1 Serial configuration

The first configuration has been inspired by the LSTM model itself: in fact,it has been demonstrated that giving larger windows, the algorithm will recognize better new unseen sequences.
To do so, I create 'virtual' time windows, concatenating the single ones belonging to different channels: the hardware implementation will be the same, changing from one to which ever number of channels; however, incrementing this number, the time to work on the virtual window will increase linearly.
Therefore, the structure of this configuration is exactly the one presented in figure 3.6.

3.3.2 Parallel configuration

For what concerns the second configuration, I try to learn separately all the channels, thinking that concatenating them all together could lead to a loss

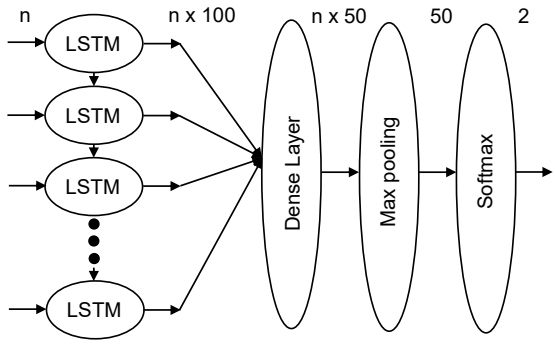

Figure 3.6: Long-short term memory architecture. The networks directly infer on raw iEEG data without any additional preprocessing stage. The two configurations used are build on top of that model, replicating it or feeding with bigger windows. The model is composed by a first LSTM layer with 100 neurons, a dense layer that reduces the dimensionality of the space to 50, a Max Pooling stage which has 50 features as output and a last softmax layer for the prevision.

of information.
This solution is exploited using a different LSTM net for each channel (figure 3.5).
With respect to the previous configuration I have a trade off between timing and hardware:

- the serial architecture has a constant memory footprint, not correlated with the number of channels: increasing the number of channel, linearly increase the computational time;
- the parallel configuration, has a constant computation time, because each channel is treated separately in a parallel fashion. However, the memory occupation grows with the number of channels.

However, to fully exploit the parallelization of that technique, I should have a number of cores equals to the number of channels: the structure is not

completly scalable.
Conversely, the serial configuration scales very well if I have a sufficient fast computation.

3.4 Convolutional Neural Network with STFT

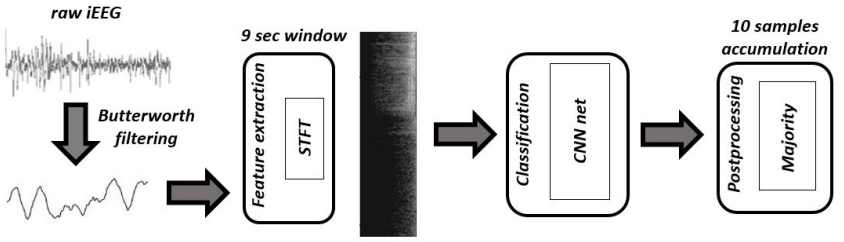

Figure 3.7: High level pipeline based on Convolutional neural network: in figure 3.9 can be found a much more detailed explanation of the network.

The convolutional neural networks work normally on 2-D inputs, as images, to extract features and achieve good classification; in this case, the highlight is on the generality of the algorithm: with respect to hand-crafted feature extractor, the CNN could automatically extract good features from images.
The CNN are the core of the deep net analysis, which contains all algorithm with many layers. They are differentiated from classical machine learnig approaches by the high number of steps used to achieve the classification.
The most important blocks of this type of net are:

1. **Convolutional layers**, used like neurons of feed-forward net, but with only a local connectivity, e.g. the 9 nearest neurons of the previous layers. A full layer is composed by n neurons, each one connected to only a portion of neurons of previous layer.

2. **Pooling layers**, used to merge informations from near neurons: the

principal type of pooling are *mean* and *max*, which takes respectively the mean and the max value of a group of neurons.

3. **non-linear layers**, whose function is comparable with the activation function at the output of a single neuron of a MLP net.

In figure 3.8 I report the convolutional layer and the pooling layers.

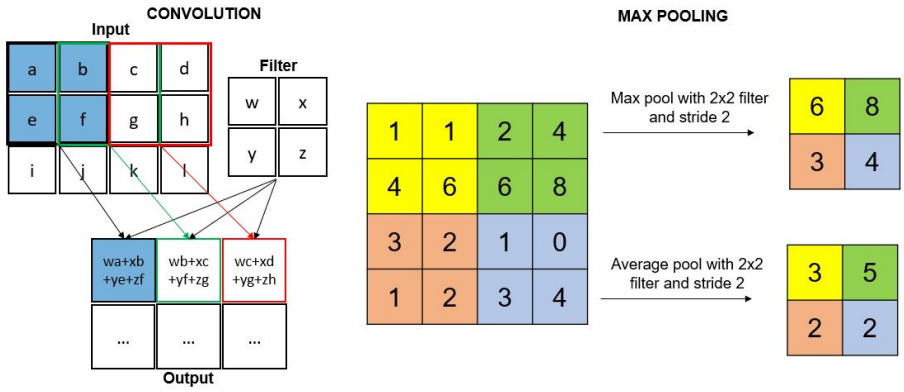

Figure 3.8: On the left a convolutional layer with a single filter and a 2D input is represented. On the right, the max and the averaging pooling for a 2D input are shown.

For my purpose, I use a predefined model of CNN, since building a CNN from scratch is expensive in terms of time (due to the high number of hyperparamethers).

The Convolutional Neural Network model has been taken from [13] and adapted for the detection problem. The CNN has three blocks and two final layers, as described in Fig. 3.9. Each block is composed by a convolution layer with rectified linear unit activation function and a max pooling. The former layer has 16 filters with 5 x 5 kernels and a stride of 2; the next ones have respectively 32 and 64 filters, with 3 x 3 kernel and stride 1. The two fully-connected layers trasform the output first to 128 and then to the

2 final outputs, ictal and interictal. Unlike the previous methods, based on LBP, the CNN needs a two-dimensional input; the short-time Fourier transform (STFT) is used to convert the raw EEG data in a two-dimensional matrix with frequency and time on axis. I used window of 9 seconds for this algorithm, since it was impossible to have a lower dimension mantaining the same structure (e.g. the strides, max pooling layers, ...).

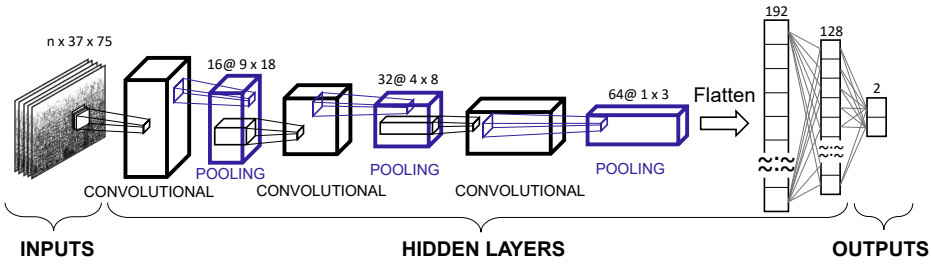

Figure 3.9: Convolutional neural network architecture. Short time Fourier transform is used to feed input with 9 seconds time-frequency windows. The model is composed by three convolution blocks. Each block consists of a convolution layer with a rectified linear unit (ReLU) activation function, and a max pooling layer. The first block is composed by 16 filters with 5 x 5 kernels, with stride of 2. Max pooling is applied over a 2 x 2 region reducing both dimensions of a factor of 2. The same steps are applied in the two consequent blocks, with 32 and 64 filters, respectively, kernel size 3 x 3 and stride 1. The features extracted are flattened and two fully connected layers, with sizes of 128 and 2, respectively, gives the final output. The former fully connected layer uses a sigmoid activation function, while the latter uses a soft-max activation function. Both of the fully connected layers have a dropout rate of 0.5.

Chapter 4

Results

In this chapter I first introduce the 2 novel datasets (Section 4.1) and then I expose the results of the seizure detection for both the datasets, togheter with the spatial analysis of the brain for a group of 11 patients.

4.1 Datasets

The datasets are created from patient of the Bern Inespital epilepsy program. Except for the need for invasive EEG studies, there were no additional inclusion criteria. All the patients gave written informed consent that their data from iEEG might be used for research purposes. The decision on the necessity for iEEG recordings, the electrode implantation scheme, and the decision on surgical therapy was made entirely on clinical grounds. These decisions were taken prior to and independently from the construction of this dataset.

iEEG signals were recorded intracranially by strip, grid, and depth electrodes (all manufactured by AD-TECH, Wisconsin, USA), using a Nicolet One recording system with a C64 amplifier (VIASYS Healthcare Inc., Madison, Wisconsin, USA). An extracranial electrode, localized between 10/20 positions Fz and Cz, was used as reference for signal recording. iEEG

recordings were either sampled at 512 or 1024 Hz, depending on whether they are recorded with less or more than 64 contacts. The iEEG recorded with more than 64 contacts are down-sampled to 512 Hz prior to further analysis for the first dataset. iEEG signals were re-referenced against the median of all the channels free of permanent artifacts as judged by visual inspection. After 16-bit analog-to-digital conversion, the data were digitally band-pass filtered between 0.5 and 150 Hz using a fourth-order Butterworth filter prior to analysis and written onto disk at a rate of 512 Hz. Forward and backward filtering was applied to minimize phase distortions.

All the iEEG recordings were visually inspected by an EEG board-certified experienced epileptologist (Kaspar Schindler) for seizure identification and exclusion of channels continuously corrupted by artifacts.

4.1.1 Short time recording dataset

The first dataset that I used in this study includes 16 patients (P1–P16) of the epilepsy surgery program of the Inselspital Bern for a total of 99 recordings. Clinical data on the patients is summarized in Table 4.1.

Each recording consists of 3 minutes of interictal segments (immediately before the seizure onset), the ictal segment (ranging from 10 s to 1002 s), and 3 minutes of postictal time. In addition to the iEEG data for each patient, the dataset includes the age, the indices of channels, the indices of resected channels, the MRI findings, the epilepsy syndrome and the post-surgical outcome. The dataset will be soon available online.

Table 4.1: Clinical characteristics of patients and seizures included in the dataset 1.
Abbreviations: Elec.: electrode, Seiz.: seizures, hem.: hemisphere, Eng.: Engel outcome, FLE: frontal lobe epilepsy, TLE: temporal lobe epilepsy, PLE: parietal lobe epilepsy, L: left, R: right, MRI: magnetic resonance imaging, y/n: yes/no

ID	**Elec.** [#]	**Seiz.** [#]	**Seizure duration** [s]			**age**[y]	**hem.**	**MRI findings**	**Syndrome**	**Eng.**
			Mean	**Min**	**Max**					
P1	100	5	14	10	22	46	R	n	TLE	2
P2	64	4	146	89	179	48	L	y(hippocampal sclerosis)	TLE	1
P3	62	14	98	31	139	32	L	y(focal cortical dysplasia)	PLE	4
P4	42	4	223	96	301	19	L	y(hippocampal sclerosis)	TLE	1
P5	59	6	88	67	117	31	L	y(hippocampal sclerosis)	TLE	2
P6	36	2	15	14	16	31	R	y(tuberous sclerosis)	FLE	4
P7	74	7	587	154	1002	36	L	y(pilocytic astrocytoma)	PLE	1
P8	61	3	121	52	184	23	L	n	TLE	4
P9	92	6	79	19	100	49	R	y(focal cortical dysplasia)	FLE	2
P10	47	13	71	10	252	24	L	n	TLE	2
P11	59	2	57	52	61	38	L	n	TLE	4
P12	54	10	99	80	154	20	R	n	TLE	2
P13	98	2	99	73	125	25	R	n	TLE	1
P14	49	10	45	23	93	59	L	y(space occupying amygdala)	TLE	4
P15	56	9	144	104	198	27	L	n	TLE	1
P16	64	2	109	83	135	26	R	y(hippocampal sclerosis)	TLE	1
Mean	**64**	**6**	**124**	**60**	**192**	**33**				

4.1.2 Long time recording dataset

Table 4.2: Clinical characteristics of patients and seizures included in the dataset 2.
Abbreviations: Elec.: electrode, Seiz.: seizures

ID	Elec. [#]	Seiz. [#]	Time of recording [h]	Mean seizure duration [s]	Mean seizure distance [m]
L1	88	2	293.36	602	9945.13
L2	66	2	235.22	88	212.68
L3	64	4	158.38	65	197.87
L4	32	14	40.70	42	154.75
L5	128	4	110.00	16.5	227.25
L6	32	8	145.99	45	835.37
L7	75	4	68.95	69.5	655.37
L8	61	4	143.79	21.94	98.60
L9	48	25	40.88	42.37	92.07
L10	32	17	42.39	71	148.35
L11	32	2	212.22	91.5	1067.37
L12	56	9	191.42	146.44	725.28
L13	64	7	103.99	102.71	857.68
L14	24	2	161.39	25.85	73.77
L15	98	2	195.92	94.5	8391.73
L16	34	5	177.05	190.8	635.85
L17	60	2	129.60	97.5	227.38
L18	42	5	205.07	199	1407.25

The second dataset that I used in this study includes 18 patients (L1–L18) of the epilepsy surgery program of the Inselspital Bern.
For each patient, a single recording of multiple hours is present. A total of 118 seizures are contained together with 2656 hours of iEEG recording.

Clinical data on the patients is summarized in Table 4.2.

Each recording consists of 40–293 hours of recording. The recording is mostly composed by interictal period, and 2–25 seizures are reported during that period. Since the recording is continuos in time, i.e. without any interruption of recording, this dataset could be used also for future experiments about prediction of seizures. The dataset will be soon available

online.

4.2 Seizure Detection Results

In this section, I compare my method to the state-of-the-art (Chapter 3) for detecting seizures in the first dataset (Section 4.2.1) and in the second dataset (Section 4.2.2).

4.2.1 Dataset 1 experimental results

I compare my algorithm (LPB+HD) with the aforementioned state-of-the-art methods using the first dataset described in Section 4.1. The results of LSTM + Linear Layer (parallel LSTM) have been neglected, as overperformed by the serial LSTM. The algorithm is implemented in Python 2.7 using PyTorch. To have an identical setup, for the state-of-the-art methods I also add the postprocessing step that is tuned for each of them to increase their specificity.

I report performance metrics including specificity, sensitivity, macroaveraging accuracy, and delay of seizure onset detection given a limited number of trained seizure examples. Sensitivity is defined as the percentage of correctly detected seizures in the test dataset. Specificity is defined as the percentage of misclassified interictal 0.5 s window. Macroaveraging accuracy is the mean of sensitivity and specificity that gives them equal weight to address the unbalancing issue between the ictal and interictal segments [59]. Latency is measured as the time that algorithm takes to classify an unseen seizure after the seizure onset time point that is marked by the expert.

Tables 4.3, 4.4, 4.5 illustrates the full comparison results. I divide all available seizures for a patient into only two sets: training and testing (no evaluation set). I train all the methods only on the training set, and measure the metrics on the test set using k-fold cross-validation, where k is the total number of seizures minus the number of seizures in the training set (i.e., I

Table 4.3: Performance of my algorithm (LBP+HD). The upper part of table shows the results in the one-shot learning setting, while the part below it shows the results of the few-shot learning.
Abbreviations and symbols used in this table: TrS: number of trained seizures, k: number of folds in cross-validation, ℓ: latency as delay of seizure onset detection, Spe.: specificity, Sen.: sensitivity, elec.: electrodes, and Accuracy: macroaveraging accuracy.

One-shot learning					
			LBP + HD Computing		
ID	**TrS**	k	ℓ [s]	**Spe.** [%]	**Sen.** [%]
P2	1	4	15.1	100	100
P4	1	4	34.5	100	100
P5	1	6	20.9	100	100
P6	1	2	6.3	100	100
P8	1	3	13.2	100	100
P11	1	2	7.0	100	100
P13	1	2	10.0	100	100
P16	1	2	32.3	100	100
mean			**17.4**	**100.0**	**100.0**
Few-shot learning					
P1	2	4	6.3	100	100
P15	2	8	36.4	100	100
P3	3	12	21.8	79.97	91.03
P7	3	5	5.0	49.9	88.57
P9	3	4	16.2	96.31	96.43
P10	3	11	3.9	98.41	94.41
P12	6	5	15.9	96.88	80
P14	4	7	10.5	95.94	85.71
mean			**14.5**	**89.68**	**92.02**
Total mean			**15.9**	**94.84**	**96.01**
Accuracy				**95.42**	

rotate the trained seizures among all available seizures).

Based on the number of seizures used for training, I observe that the patients may be roughly partitioned into two groups: patients with one-shot

Table 4.4: Comparisons of my algorithm (LBP+HD) with the state-of-the-art methods: LBP+SVM [11], LGP+MLP [11]. The upper part of table shows the results in the one-shot learning setting, while the part below it shows the results of the few-shot learning.
Abbreviations and symbols used in this table: TrS: number of trained seizures, k: number of folds in cross-validation, ℓ: latency as delay of seizure onset detection, Spe.: specificity, Sen.: sensitivity, elec.: electrodes, and Accuracy: macroaveraging accuracy.

One-shot learning								
			LBP + Linear SVM			**LGP + MLP**		
ID	**TrS**	k	ℓ [s]	**Spe. [%]**	**Sen. [%]**	ℓ [s]	**Spe. [%]**	**Sen. [%]**
P2	1	4	10.1	91.74	75	12.2	98.26	100
P4	1	4	29.3	100	100	35.2	100	100
P5	1	6	14.7	92.09	100	14.6	84.54	100
P6	1	2	9.0	100	100	7.5	100	100
P8	1	3	11.9	100	100	10.3	100	100
P11	1	2	6.5	100	100	6.5	100	100
P13	1	2	16.3	100	100	9.8	100	100
P16	1	2	29.3	100	100	29.5	96.81	100
mean			**15.9**	**97.98**	**96.88**	**15.7**	**97.45**	**100.0**
Few-shot learning								
P1	2	4	6.9	100	100	6.9	96.76	100
P15	2	8	31.3	99.86	100	30.8	91.71	100
P3	3	12	15.6	81.33	100	16.8	77.04	100
P7	3	5	5.0	51.78	88.57	10.9	65.53	91.42
P9	3	4	12.4	89.05	96.42	14.4	88.93	89.28
P10	3	11	9.2	97.39	92.31	13.3	98.78	88.81
P12	6	5	13.6	82.62	88	17.3	85.01	90
P14	4	7	9.1	98.02	95.71	9.9	94.58	94.28
mean			**12.9**	**87.51**	**95.13**	**15.0**	**87.29**	**94.22**
Total mean			**14.4**	**92.74**	**96.00**	**15.4**	**92.37**	**97.11**
Accuracy				**94.37**			**94.74**	

learning (in the upper part of Table 4.3), and patients that need few more training seizures (few-shot learning) in the lower part of the table. For half

Table 4.5: Comparisons of my algorithm (LBP+HD) with the state-of-the-art methods: LSTM [12], STFT+CNN [13] The upper part of table shows the results in the one-shot learning setting, while the part below it shows the results of the few-shot learning.
Abbreviations and symbols used in this table: TrS: number of trained seizures, k: number of folds in cross-validation, ℓ: latency as delay of seizure onset detection, Spe.: specificity, Sen.: sensitivity, elec.: electrodes, and Accuracy: macroaveraging accuracy.

One-shot learning								
			LSTM			**STFT + CNN**		
ID	**TrS**	k	ℓ [s]	**Spe.** [%]	**Sen.** [%]	ℓ [s]	**Spe.** [%]	**Sen.** [%]
P2	1	4	10.6	100	100	11.2	95.63	100
P4	1	4	29.5	100	100	31.3	100	100
P5	1	6	17.1	99.31	100	26.3	100	100
P6	1	2	8.9	100	100	15.0	100	100
P8	1	3	10.9	100	100	18.5	100	100
P11	1	2	7.08.5	100	100	12.8	100	100
P13	1	2	11.3	100	100	15.8	100	100
P16	1	2	27.3	100	100	35.0	100	100
mean			**15.5**	**99.91**	**100**	**20.7**	**99.45**	**100.0**
Few-shot learning								
P1	2	4	5.9	99.98	100	15.2	99.07	100
P15	2	8	31.4	99.99	100	45.1	100	100
P3	3	12	16.1	86.85	55.30	13.65	67.07	63.6
P7	3	5	14.1	50.16	85.00	4.7	16.7	100
P9	3	4	7.5	82.98	100	13.2	86.95	100
P10	3	11	8.9	98.67	99.09	14.9	98.36	100
P12	6	5	17.6	99.56	90	6.8	35.00	100
P14	4	7	9.6	98.85	92.86	6.75	39.03	100
mean			**13.9**	**89.63**	**90.28**	**15.0**	**67.77**	**95.45**
Total mean			**14.7**	**94.77**	**95.14**	**17.9**	**83.61**	**97.73**
Accuracy				**94.96**			**90.67**	

of the patients (8 out of 16), my algorithm exhibits one-shot learning, i.e. training with only one seizures is possible. My algorithm achieves perfect

(100%) specificity and sensitivity in detecting novel seizures. The other methods cannot exhibit such perfect generalization in the one-shot setting: the LSTM is the closest method that achieves 99.91% specificity. Overall, all the methods achieve better performance for these patients in the upper part of table.

The remaining eight patients, listed in the lower part of Table 4.3, are more challenging due to their fast and very localized seizures (i.e., only 2 or 3 electrodes out of 70 are involved in the ictal activity). Hence for these patients few more seizures (2–6) are required for training. my algorithm trained with two seizures still maintains the perfect generalization for two more patients (P1 and P15) while the other methods are behind. Only for two patients (P3 and P7), my algorithm shows a low specificity in the few-shot learning; on average, it achieves 89.68% specificity (vs. 89.63% in the LSTM) and 92.02% sensitivity (vs. 90.28% in the LSTM).

It is worth to briefly discuss two types of variation that may occur for seizures. The seizures may start focal and remain focal (i.e., restricted to one lobe/hemisphere) or they may secondary generalize and involve both cerebral hemispheres. Importantly for this type of variation the iEEG patterns emerging at the seizure onset are very similar and thus seizure onset should be rapidly detected. On the other hand, few patients may have seizures starting in different regions of the brain, for example some seizures begin in the left, some in the right temporal lobe (so-called bilateral temporal lobe epilepsy). If this is the case, there will be different iEEG patterns at seizure onset for the different types of seizures, and both types have to be learned by the algorithms.

Considering both one-shot and few-shot settings across Table 4.3, my algorithm, on average, achieves higher macroaveraging accuracy (95.42%) than the other methods. The LSTM is the closest method that reaches 94.96% macroaveraging accuracy thanks to the added postprocessing method to avoid otherwise lower accuracy of 90.46%. Although other methods reach

slightly higher sensitivity for some of the patients, my algorithm achieves the highest specificity (94.84% on average) that clearly shows the limitation of the other methods for long-time recordings. The latency of seizure onset detection of my algorithm is slightly larger than the one yielded by the LSTM (15.9 s vs. 14.7 s). Given that the average duration of ictal segments is 123.6 s (see Table 4.1), my algorithm still detects the seizures in the first 8% of the ictal window. Such a delay of detection below 20 s is well suited for several important applications considering that iEEG seizure onset often precedes clinical onset by more than 20 s [60]. Furthermore, considering that the median durations of mesial temporal lobe seizures and of neocortical extratemporal seizures have been found to be 106 s and 78 s, respectively [61], it may be helpful for theraupeutic interventions aimed a early seizure termination or prevention of generalization of seizure activity [62].

More importantly, my algorithm operates with simple binary operations and a lower memory footprint. As the output of training, my algorithm requires to store only the contents of the AM: $2*d$ bits for the two prototypes. The IM can be efficiently *rematerialized* by a cellular automa from a random seed [63], hence there is no need to store the IM [37].

Table 4.6: The table shows the memory requirement to store each model (i.e. the learned weights).

	Memory footprint				
	HD	**SVM**	**MLP**	**LSTM**	**CNN**
Weights [#]	2496	256 × # e.	10240 × # e. + 160	185408	1600 × # e. + 194056
	1	4–10×	148–410×	74×	101–142×

The Table 4.6 shows the number of weights to be stored for each model. The number of weights for all the state-of-the-art methods (except the LSTM) grows with the number of electrodes. Though the number of weights in the LSTM are constant, its computational time grows with the number

of electrodes, since it is a sequential model. Considering d=10,000, my method is stored on 2,500 bytes resulting in at least 4–10× lower memory requirements with respect to the simplest SVM classifier, and up to 74× with respect to the LSTM. Moreover, this is a conservative memory estimation for my algorithm as the dimensionality can be reduced to 1000 for some iEEG recordings without affecting the performance. And finally, my algorithm works with binary operations while other methods employ floating-point operations.

4.2.2 Dataset 2 experimental results

In this section, I propose again the comparison of my algorithm (LPB+HD) with the aforementioned state-of-the-art methods using the second dataset described in Section 4.1. I decide to neglect LSTM+Linear Layer and MLP+LGP results, because strongly overperformed by the other soa methods.

As an extension of previous experiments, for the state-of-the-art methods I also add the postprocessing step that is tuned for each of them to increase their false detections per hour(FpH). Contrary to the previous set of experiments, removing the postprocessing step in unfeasible since the FpH will grow so much, limiting a real implementation. In fact, multiple false alarm could cause a higher level of anxiety, that could cause an increasing of the epileptic activity [64].

I report performance metrics including FpH, sensitivity, and delay of seizure onset detection given a limited number of trained seizure examples. Sensitivity is defined as in the previous experiments. FpH is defined as the number of false alarms occurred during an hour. Two false positive occurring nearer than 10 minutes one from the other are classified as the same false seizure. Latency is again measured as the time that algorithm takes to classify an unseen seizure after the seizure onset time point that is marked by the expert. Table 4.7 illustrates the full comparison results.

Table 4.7: Abbreviations and symbols used in this table: FDR: false detection rate, ℓ: latency as delay of seizure onset detection, Sen.: sensitivity, El.: electrodes, Sz.: seizures in the recording, and Hours: duration of the recording in hours, n.a. not applicable.

		LBP + HD Computing				**LBP + Linear SVM**			**LSTM**			**STFT + CNN**		
ID	**TrS.** [#]	ℓ [s]	**FDR.** [/h]	**Sen.** [%]	d [kb]	ℓ [s]	**FDR.** [/h]	**Sen.** [%]	ℓ [s]	**FDR.** [/h]	**Sen.** [%]	ℓ [s]	**FDR.** [/h]	**Sen.** [%]
P1	1	11.0	0.00	100.0	3	10.0	0.00	100.0	8.0	0.10	100.0	8.0	0.00	100.0
P2	1	15.0	0.00	100.0	10	8.0	0.75	100.0	17.0	0.40	100.0	3.0	0.75	100.0
P3	1	12.7	0.00	100.0	7	7.0	0.05	100.0	5.7	0.20	100.0	2.0	0.00	100.0
P4	2	19.9	0.00	71.4	6	30.0	0.65	50.0	24.3	1.20	91.7	22.5	0.00	41.7
P5	1	3.0	0.00	100.0	1	2.7	0.25	100.0	5.7	0.30	100.0	1.3	0.15	100.0
P6	1	9.8	0.00	85.7	10	10.0	0.20	85.7	12.3	0.20	100.0	0.8	1.90	85.7
P7	2	16.5	0.00	50.0	1	26.5	1.15	50.0	8.3	1.45	100.0	26.0	0.00	100.0
P8	2	18.3	0.00	100.0	10	2.0	1.30	100.0	8.5	1.05	100.0	16.3	1.20	100.0
P9	1	3.8	0.00	95.8	6	16.3	0.10	41.7	30.0	0.05	20.8	n.a.	0.00	0.0
P10	1	9.6	0.00	100.0	3	3.6	0.10	100.0	25.8	1.60	100.0	39.8	1.00	93.8
P11	1	19.4	0.00	100.0	3	12.0	0.40	100.0	7.0	0.05	100.0	5.0	0.20	100.0
P12	2	36.0	0.00	100.0	1	27.6	0.00	100.0	28.4	1.15	100.0	10.9	0.00	100.0
P13	1	37.7	0.00	100.0	2	11.3	0.00	100.0	6.2	0.90	100.0	1.3	0.40	100.0
P14	1	n.a.	0.00	0.0	1	n.a.	0.00	0.0	n.a.	0.00	0.0	n.a.	0.00	0.0
P15	1	15.0	0.00	100.0	1	3.0	0.15	100.0	2.0	0.05	100.0	5.0	0.00	100.0
P16	1	12.8	0.00	100.0	10	9.0	0.55	100.0	8.3	0.80	100.0	7.0	0.20	100.0
P17	1	19.0	0.00	100.0	1	13.0	0.00	100.0	3.0	0.10	100.0	16.0	0.45	100.0
P18	1	34.0	0.00	75.0	1	26.3	0.00	75.0	18.8	0.15	100.0	14.7	0.20	75.0
mean		**17.3**	**0.00**	**87.7**	**4.3**k	**12.8**	**0.31**	**83.5**	**12.9**	**0.54**	**89.6**	**11.2**	**0.36**	**83.1**

The experiments are divided into two sets: for the main algorithm, the HD + LBP, I use a window of 30 seconds interictal iEEG and 1 or 2 seizures to train and I test the full recording, reporting number of false alarms, number of seizure detected and delay; for the state-of-the-art competitors, I use the same train setup, but I tested on 20 hours of recording of interictal iEEG and on all the fragments containing a seizure, because limited by the computational resources. I use 1 seizure for training for 14 of 18 patients and 2 seizures for the other 4 (L4, L7, L8, L12). No evaluation set has been considered.

For the 2/3 of the patients (12 out of 18), my algorithm exhibits ideal performances, in terms of Sensitivity (100%) and false detections (0). For all the patients (18 out of 18), my algorithm achieves an impressive 0.0 false detection rate, avoiding any false seizure detection for a total of 2656 hours of recordings.

The other methods cannot exhibit such perfect generalization: the SVM is the closest method that achieves 0.31 FpH. Overall, all the competitors achieve comparable performances in term of Sensitivity with the HD algorithm, but with a higher FpH (0.31–0.54). Only for two patients (L7 and L14), my algorithm shows a very low sensitivity in the few-shot learning (50% and 0%); noteworthy, for patient L14, all the methods fail in the detection of the unseen seizure. This low sensitivity could be linked to different types of variation that may occur for seizures, as previously described (Section 4.2.1). On average, my algorithm achieves 87.67% sensitivity (vs. 89.59% in the SVM) and 0.0 FpH (vs. 0.54 in the SVM). The latency of seizure onset detection of my algorithm as for the other dataset is slightly larger than the one yielded by the competitors (17.26 s vs. 12.84 s of SVM). The considerations done for the previous dataset hold, yielding the HD latency to be good enough for several applications (e.g. closed loop seizure termination).

Finally, I further investigate the dimension (d) of hypervectors. For the

experiments on short-time dataset (Section 4.2.1), I use a constant dimension of 10000; conversely, for this dataset I tuned the dimension per patient, reducing it till there aren't worsening of the performance. In Table 4.7 the tuned dimension is reported for each patient. As we can see, almost half of the patients shows the same performances with $d = 1000$. Reducing d, I can save memory footprint (10× higher gain), power consumption and computational time.

4.3 Spatial analysis results

In this section, I report the identification accuracy and the p-value of my algorithms for determining the ictogenic regions. For identifying the ictogenic hemisphere, I assess how my algorithm performs on the patients with bilaterally localized electrodes (11 out of 16 patients; see Fig 2.10). For these patients, ictogenic hemispheres are indicated by the experts as previously reported in [5]. The second column of Table 4.8 lists these hemispheres as ground truth, and shows the mean score of my algorithm for the left and right hemispheres. When the score of one side significantly differs from the other side ($p < 0.01$), the algorithm classifies the hemisphere with the highest score as the ictogenic hemisphere. Using one or two seizures for training, the algorithm correctly classifies the ictogenic hemisphere with an accuracy of 100% (11 of 11) and $p < 0.01$.

Four patients out of 11 require at least two seizure examples for robust identification, otherwise their p-values will be higher than 0.01. Moreover, for nine patients with more recorded seizures, I perform further analysis by increasing the number of training seizures (see the second half of Table). The p-value decreases for eight patients thus implying a more robust identification. Noteworthy for five patients, the reduction is higher than two order of magnitude. However, for P15, the tests show a higher p after training with all the seizures. This might be due to the strongly asymmetric

Table 4.8: Patients with bilaterally localized electrodes are analyzed to identify their ictogenic hemisphere. The first part shows the minimum number of seizures for training (TrS.) to identify the ictogenic hemisphere with $p < 0.01$; also the mean scores of the hemispheres and the p-value of the t-test. The experiments are also repeated by training with all available seizures.

		Trained with min. # of seizures s.t. $p < 0.01$				**Trained with all available seizures**			
ID	**Hem.**	**TrS. [#]**	**Mean Score Left Hem.**	**Mean Score Right Hem.**	p**-value**	**TrS. [#]**	**Mean Score Left Hem.**	**Mean Score Right Hem.**	p**-value**
P1	R	1	0.1176	0.1765	0.00005	5	0.1091	0.1754	< 0.000001
P2	L	1	0.2531	0.1092	0.00001	4	0.2481	0.0901	< 0.000001
P5	L	1	0.3496	0.2216	0.002	6	0.3466	0.2355	< 0.000001
P6	R	2	0.0966	0.1608	0.005	2	0.0966	0.1608	0.005
P8	L	1	0.1057	0.0617	0.0005	3	0.1034	0.0741	0.0002
P9	R	2	0.1010	0.1327	0.0009	6	0.0919	0.1372	< 0.000001
P11	L	1	0.1649	0.1040	0.008	2	0.2064	0.1512	0.003
P12	R	1	0.0549	0.0778	0.004	10	0.0615	0.0931	< 0.000001
P13	R	2	0.1555	0.2136	0.0001	2	0.1555	0.2136	0.0001
P15	L	2	0.2510	0.1675	0.00007	9	0.2693	0.2175	0.0008
P16	R	1	0.2342	0.3704	0.00002	2	0.2393	0.3445	0.00001

implantation scheme for this patient.

For identifying the ictogenic lobe, I use the data from two patients for whom the exact SOZ localization is available [5]. The algorithm correctly localizes the SOZ for only one of them. This result can be compared to the post-surgical outcome: the algorithm correctly classifies the SOZ for the patient who remained seizure free after surgery (3 years follow-up), whereas the other one did not improve, thus suggesting that the SOZ was more extensive than assessed during pre-surgical evaluation. These findings are promising but further studies are warranted to confirm my results.

Chapter 5

TX2 implementation

In this chapter I will report an efficient implementation of the algorithm on the Tx2 platform. The Tx2 platform offers an embedded platform to run algorithms in a linux environment. I use the Tx2 platform to make a fair comparison of energy consumption and time of execution of all the methods aforementioned.

In the following, I will present the Jetson TX2 platform (Section 5.1), an efficient parallelization of the HD algorithm on the GPU (Section 5.2), and the comparation among all the algorithms in term of energy and time of execution (Section 5.3).

5.1 Jetson TX2 computing device

Jetson Tx2 is a powerefficient embedded AI computing device. Jetson TX2 is built around an NVIDIA Pascalfamily GPU and loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth. It features a variety of standard hardware interfaces that make it easy to integrate it into a wide range of products and form factors.

The Jetson TX2 module integrates:

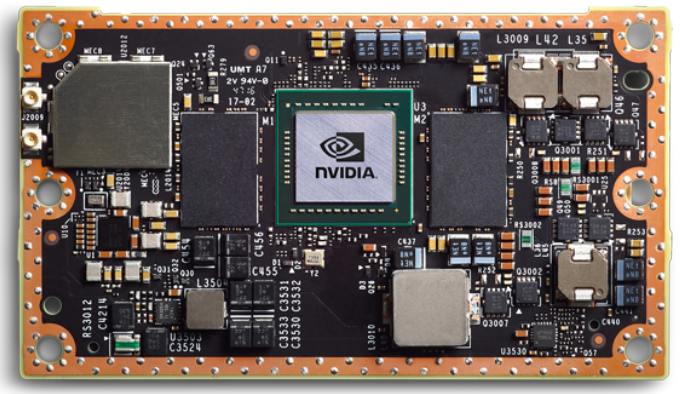

Figure 5.1: Jetson Tx2 platform, an AI supercomputer features NVIDIA Maxwell architecture, 256 NVIDIA CUDA cores, 64-bit CPUs, and a power-efficient design.

- **256 core NVIDIA Pascal GPU**. Fully supports all modern graphics APIs, unified shaders and is GPU compute capable. The GPU supports all the same features as discrete NVIDIA GPUs, including extensive compute APIs and libraries including CUDA. Highly power optimized for best performance in embedded use cases.

- **RMv8 (64-bit) Multi-Processor CPU Complex**. Two CPU clusters connected by a high-performance coherent interconnect fabric designed by NVIDIA; enables simultaneous operation of both CPU clusters for a true heterogeneous multi-processing (HMP) environment. The Denver 2 (Dual-Core) CPU clusters is optimized for higher single-thread performance; the ARM Cortex-A57 MPCore (Quad-Core) CPU clusters is better suited for multi-threaded applications

and lighter loads.

- **8GB LPDDR4** and **32 GB eMMC** memory integrated on the module.

- **128-bit Memory Controller**.128-bit DRAM interface providing high bandwidth LPDDR4 support.

It also includes an advanced HD Video Encoder, an HD Video Decoder, a Display Controller Subsystem, a 1.4Gpix/s Advanced image signal processing and Audio Processing Engine.

Moreover, the TX2 board includes sensors on the board and the module to perform power measurements at different levels of granularity. Reading out these sensors allows to perform power measurements on the TX2 board.

Finally, the TX2 board could perform operations in different modes, to save power or to achieve best performances. Using the command *$ sudo nvpmodel -m [mode]* the TX2 board could work with the different modes reported in table 5.1. The TX2 could work with both the CPU cluster and the GPU at max frequency (Max-N) to optimize speed or with the ARM cortex only and the GPU at minimum frequency (Max-Q) to save power. All the other modes are in between, as trade-off between speed and power saving.

Table 5.1: Different modes of operation of TX2 platform. Max-N achives the highest speed of execution, Max-Q the maximum power saving.

Mode	Mode Name	Denver 2	fs	ARM A57	fs	GPU fs
0	Max-N	2	2.0 GHz	4	2.0 GHz	1.30 GHz
1	Max-Q	0		4	1.2 GHz	0.85 GHz
2	Max-P Core-All	2	1.4 GHz	4	1.4 GHz	1.12 GHz
3	Max-P ARM	0		4	2.0 GHz	1.12 GHz
4	Max-P Denver	2	2.0 GHz	0		1.12 GHz

5.2 HD Acceleration on GPU

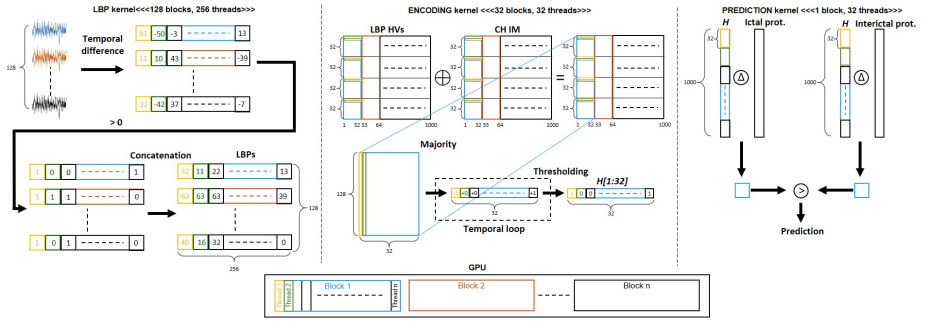

Figure 5.2: Parallelization of the algorithm on a GPU. For each kernel, 2 threads and 2 blocks are highlighted to show the distribution of the operations inside the GPU.

In this section, I will highlight the most important step of the parallelization of the algorithm on the GPU of the TX2 device. I exploit the CUDA code to ensure a full employment of all the CUDA cores.

The code is composed of three CUDA kernels, that perfectly match the three sections of the algorithm. In the following, I describe the structure of each kernel, using 128 as number of electrodes:

- **LBP kernel**: this kernel is used to compute the LBP of a time window. *Inputs*: EEG signal; *Outputs*: LBPs matrix; *Blocks/threads*: 128/256; each single block of the GPU analyses a different channel, since the computation of LBP is channel specific and is uncorrelated with the LBP of other channels.

  Each thread inside the block, first move a sample of the EEG data to the shared memory to speed up the computation, then calculates the temporal difference between the corresponding sample and the next one, and finally creates the LBP associated to that point.

- **ENCODING kernel**: this kernel trasforms the LBPs matrix into

a single vector to be compared with the associative memory. *Inputs*: LBPs matrix, LBPs' IM, channel's IM; *Outputs*: encoded vector; *Blocks/threads*: 32/32; for this kernel, each block takes into account only 32 bits of both the IM and produce 32 bits of the encoded vector. For example, block 0 uses bits 0-31 of the IMs and produces the bits 0-31 of the output; block 1 uses bits 32-63 and produces the bits 32-63 of the output.

Conversely, at the threads level we have two different steps: first, each thread copies to the shared memory 4 vectors of the channel's IM (i.e. 4 channels) and 2 vectors of the LBPs' IM. In this way, both the memory are completely copied to the shared memory. After, each thread works on one bit. In this phase, each thread performs the majority across the 128 channels, generating a new bit for each time sample. Finally, the thread performs again the majority among the full time window (256 samples). The output is a $32 * 32$ bit vector that encodes the time window of 256 samples.

- **PREDICTION kernel**: this kernel is used to give the label to the test window. *Inputs*: query vector, associative memory, and predictions' vector; *Outputs*: label of the window; *Blocks/threads*: 1/32; this kernel is simpler than the previous ones. Each thread takes into account 32 bits of all the inputs, performing xor between query and both the label vectors and then the popcount. The two hamming distances are computed summing up all the threads results. Finally, the postprocessing is applied by a single thread that computes the window label.

In Fig. 5.2 all the kernels are visually described.

Furthermore, the dimension of the encoded window could be tuned as 0.5 second or 1 second. Only applying the 3 kernels, a window of 0.5 second is used to produce the encoded vector. Conversely, if you want to change

from 0.5 seconds to 1 second (0.5 seconds has been applied to first dataset, 1 second to the second one), you simply compose the encoded vector with the one of the previous window before applying the *PREDICTION kernel*, keeping information from two successive windows of 0.5 seconds, i.e. 1 second.

5.3 Experimental Results and scalability

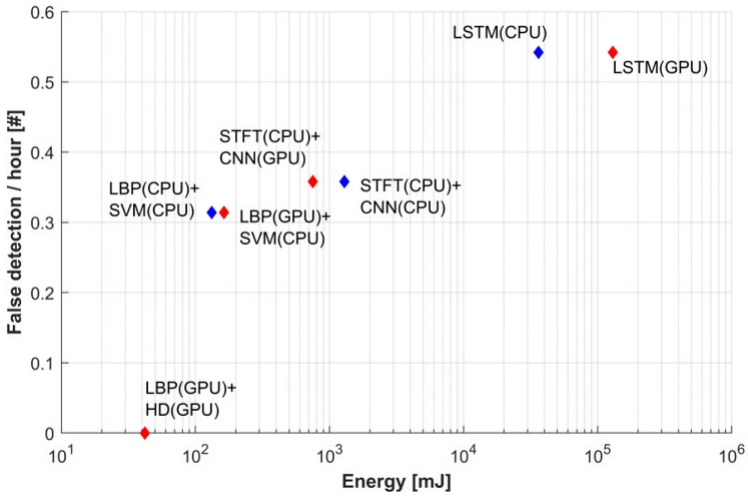

Figure 5.3: Comparison of my algorithm with state-of-the-art methods in terms of energy and false detection / hour. Energy per classification is plotted on the x-axis, mean false detection/hour on the 18 patients is on the y-axis.

My experiments involve running the above discussed methods on CPU clusters and on the GPU of the NVIDIA Jetson TX2. The experiments are performed with an identical setup for all the considered methods and different combination of resources (GPU and CPU) is used for each of them. I use $d = 1000$ in my algorithm and I evaluate the performance with 24 and 128 channels (respectively the minimum and the maximum number of

Table 5.2: TX2 performances of all the algorithms. The best resources (i.e. CPU and GPU) are chosen for all of them. I report time for a single classification, energy per classification, and the ratio between the state-of-the-art energy/time and my algorithm performances. I repeat the experiments in max performance mode (MAXN) and in maximum power saving mode (MAXQ). These data are drawn from the patient with the maximum number of channels, 128.

MAXN				
128 elec	HD	SVM(CPU)	CNN(GPU+CPU)	LSTM(CPU)
time[ms]	10.0	25.5	147.0	5278.0
energy[mJ]	42.0	132.6	751.0	36286.0
time gain	1.0×	2.6×	14.7×	527.8×
energy gain	1.0×	3.2×	17.9×	864.0×
MAXQ				
128 elec	HD	SVM(CPU)	CNN(GPU+CPU)	LSTM(CPU)
time[ms]	13.0	51.0	213.0	6333.0
energy[mJ]	35.0	103.0	556.0	16224.0
time gain	1.0×	3.9×	16.4×	487.2×
energy gain	1.0×	2.9×	15.9×	463.5×

Table 5.3: Same data of the previous table. The data are taken with the patient with lowest number of the channels, 24.

MAXN				
24 elec	HD	SVM(CPU)	CNN(GPU+CPU)	LSTM(CPU)
time[ms]	8.6	11.0	42.0	983.0
energy[mJ]	32.5	53.0	184.0	6588.0
time gain	1.0×	1.3×	4.9×	114.3×
energy gain	1.0×	1.6×	5.7×	202.7×
MAXQ				
24 elec	HD	SVM(CPU)	CNN(GPU+CPU)	LSTM(CPU)
time[ms]	12.5	20.8	53.0	1416.0
energy[mJ]	32.0	44.8	131.0	3980.0
time gain	1.0×	1.7×	4.2×	113.3×
energy gain	1.0×	1.4×	4.1×	124.4×

channels for patients in my dataset).

My algorithm is implemented with CUDA toolkit to maximally exploit the componentwise operations parallelization. The state-of-the-art methods are implemented using Python 3 optimized libraries (in particular keras and scikit-learn).

Fig. 5.3 depicts false detection rate and energy measurements of my algorithm using different TX2 power modes, comparing them with the state-of-the-art methods. In Table 5.3 and Table 5.2 I report the results for the best implementation of each method. I report implementation with CPUs, GPUs, or a combination of them, basing on the lowest energy consumption. Both time and energy consumption are evaluated for a single classification, that is repeated every 0.5 seconds.

Initially, I validate the methods on the board with a 30 minutes interictal segment and one seizure. The training of the model is done off-line and the trained model is loaded on the board.

Then, I evaluate execution time and energy consumption with Max-N and Max-Q mode. In a 24-channel model, my algorithm achieves 8.6 ms time of execution and 32.5 mJ energy consumption in Max-N mode, and 12.5 ms and 32.0 mJ in Max-Q mode. Noteworthy with an ASIC implementation I can further decrease the frequency of execution, increasing the time to 500 ms (real time constraint) and consistently reducing the energy consumption. Table 5.3 shows that HD roughly gains 1.3×–1.7× in time and saves 1.4×–1.6× energy compared to SVM model. Moreover, HD achieves 4.2×–114.3× speed up and 4.1×–202.7× energy saving compared to deep-learning approaches.

I also evaluate the scalability of the algorithm, increasing the number of channels from 24 to 128. As detailed in Table 5.2 the speed up increases to 2.6×–3.9× with SVM and to 14.7×–527.8× with deep-learning methods The energy saving reaches a minimum of 15.9× in comparison with the CNN. These results highlight a constant execution time and energy consumption

in the HD with a variable number of channels (e.g. Max-q mode, 12.5 s vs 13 s time, 32.0 mJ vs 35 mJ energy consumption). Conversely, the state-of-the-art methods display a considerable increment in both the metrics increasing the number of the channels from 24 to 128 (e.g. Max-q mode, 20.8 s vs 51 s time, 44.8 mJ vs 103 mJ energy consumption).

My findings show that the proposed algorithm (HD+LBP) demands for lower energy in comparison with all the other analysed methods on an embedded device as TX2. Further studies are warranted to address the energy consumption of the algorithm on an implantable device with more limited of TX2 platform.

Chapter 6

Conclusions and Future Work

My algorithm learns from one or few iEEG seizure recordings; it exploits LBP codes and HD computing that enable full binary operations during training and inference. Its learning procedure is transparent and thus allows to translate the learned codes into information about the spatial localization of the seizure-generating brain regions, for example to better target surgical resection or neuro-modulatory interventions. The algorithm perfect generalizes on a second long time dataset, showing a 0.0 false detection rate. My algorithm also provides a universal and linearly scalable interface with a minimal set of parameters that ease analyzing all iEEG recordings from different patients with 24 to 128 implanted electrodes. Using the first dataset with 16 patients and 99 seizures, my algorithm requires a total of 34 seizures for training: eight seizures for eight patients (with one-shot learning) and 26 seizures for the other eight patients (with few-shot learning). I test the algorithm on 65 unseen seizures using k-fold cross-validation: the algorithm outperforms LBP+SVM, LGP+MLP, LSTM and STFT+CNN with higher specificity and macroaveraging accuracy, and a lower memory

footprint.

I confirm these results with the second dataset: on 18 patients and 120 seizures, the algorithm again outperform the state-of-the-art, training with 22 seizures and testing on the remaining 98. The proposed algorithm consumes less energy and requires less time to perform a classification compared to the baseline.

Future work will follow in two directions:

- **Software**: it is possible to apply the described positional binding to the algorithm and to use different kinds of encoding. It could be interesting to take into account in the encoding the correlation between channels, that has been demonstrated to be connected to seizure activity [65].
- **Hardware**: focus on efficient hardware implementation on an ASIC, with a dedicated hardware for specific HD operations (distributed xor and popcount). A possible future application of the algorithm could be the development of a closed-loop seizure termination system on an implantable device.

Furthermore, the algorithm could be used to assist doctors in the tedious work of iEEG analysis and in clinical daily activity.

Appendix A

Task Description

Master Project at the Department of Information Technology and Electrical Engineering

Spring Semester 2018

One-shot Learning for Seizure Detection and Identification of Epileptogenic Brain Regions from Long-time Human iEEG Recording with End-to-end Binary Operations

Alessio Burrello

March 1, 2018

Advisor: Abbas Rahimi, ETZ J65, abbas@ee.ethz.ch
Handout March 1, 2018
Due: August 28, 2018

0.1 Introduction

The way the brain works suggests suggest that rather than working with numbers that we are used to, computing with hypervectors (high-dimensional vectors, e.g., 10,000 bits) is more efficient. HD computing offers a general and scalable model of computing as well as well-defined set of arithmetic operations that can enable fast and one-shot learning (no need of back-propagation like in neural networks). Furthermore it is memory-centric with embarrassingly parallel operations and is extremely robust against most failure mechanisms and noise. There have been successful applications in variety of tasks such as: language recognition, text classification, biosignal processing (EMG/EEG), scene reasoning, analogical-based reasoning, etc.

Hypervectors are high-dimensional (e.g., 10,000 dimensions), they are (pseudo) random with independent identically distributed components and holographically distributed (i.e., not microcoded). Hypervectors can use various coding: dense or sparse, bipolar or binary and can be combined using arithmetic operations such as multiplication, addition, and permutation. The vectors can be compared for similarity using distance metrics.

In this project, the goal would be to develop an high level algorithm exploiting HD computing for seizure detection, using intracranial EEG signals.

0.2 Project Description

There are four main objective for this project:

1. designing and developing a set of feature extractors and HD classifiers for the seizure detection/prediction problem;
2. developments of such algorithms in high-level Matlab/Python;
3. implementation in C and embedded processors.

0.3 Goals

The main goals of the project are outlined below:

- find one or multiple preprocessing algorithm in time or frequency domain, to extract good features from iEEG;
- create a full pipeline (preprocessing + classifier) to achieve good accuracy on a novel dataset;
- implement the algorithm on an embedded processor to calculate energy consumption and see real time execution;

Additional bonus goals:

- speed up the algorithm using CUDA computing, exploiting GPU with python;
- create a working version of the algorithm using the CUDA toolkit, parallelizing the HD on an embedded GPU;
- explore the state-of-the-art to find good baseline and assess their performance on the same new dataset.

0.4 Project Realization

0.4.1 Project Plan

Within the first week of the project you will be asked to prepare a project plan. This plan should identify the tasks to be performed during the project and sets deadlines for those tasks. The prepared plan will be a topic of discussion of the first week's meeting between you and your advisers. Note that the project plan should be updated constantly depending on the project's status.

0.4.2 Meetings

Weekly meetings must be held between the student and the assistants. In case a meeting cannot be done, a weekly writeup report must be written. The exact time and location of these meetings will be determined within the first week of the project in order to fit the students and the assistants schedule. These meetings will be used to evaluate the status and progress of the project. Beside these regular meetings, additional meetings can be organized to address urgent issues as well.

0.4.3 Report

Documentation is an important and often overlooked aspect of engineering. One final report has to be completed within this project. The common language of engineering is de facto English. Therefore, the final report of the work is preferred to be written in English. Any form of word processing software is allowed for writing the reports, nevertheless the use of LATEX or any other vector drawing software (for block diagrams) is strongly encouraged by the IIS staff.

Final Report The final report has to be presented at the end of the project and a digital copy need to be handed in. Note that this task description is part of your report and has to be attached to your final report.

0.4.4 Presentation

There will be a presentation (20 min presentation and 5 min QA) at the end of this project in order to present your results to a wider audience. The exact date will be determined towards the end of the work.

Bibliography

[1] F. Mormann, R. G. Andrzejak, C. E. Elger, and K. Lehnertz, "Seizure prediction: the long and winding road," *Brain*, vol. 130, no. 2, pp. 314–333, 2007. [Online]. Available: http://dx.doi.org/10.1093/brain/awl241

[2] D. Schmidt and M. Sillanpää, "Evidence-based review on the natural history of the epilepsies." *Current opinion in neurology*, vol. 25 2, pp. 159–63, 2012.

[3] J. F. TÃ©llez-Zenteno, R. Dhar, and S. Wiebe, "Long-term seizure outcomes following epilepsy surgery: a systematic review and meta-analysis," *Brain*, vol. 128, no. 5, pp. 1188–1198, 2005. [Online]. Available: http://dx.doi.org/10.1093/brain/awh449

[4] S. Wiebe, W. T. Blume, J. P. Girvin, and M. Eliasziw, "A randomized, controlled trial of surgery for temporal-lobe epilepsy," *N. Engl. J. Med.*, vol. 345, no. 5, pp. 311–318, Aug 2001.

[5] C. Rummel, E. Abela, R. G. Andrzejak, M. Hauf, C. Pollo, M. Muller, C. Weisstanner, R. Wiest, and K. Schindler, "Resected brain tissue, seizure onset zone and quantitative eeg measures: Towards prediction of post-surgical seizure control," *PLOS ONE*, vol. 10, no. 10, pp. 1–26, 10 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0141023

[6] Y. Nagahama, A. J. Schmitt, D. Nakagawa, A. S. Vesole, J. Kamm, C. K. Kovach, D. Hasan, M. Granner, B. J. Dlouhy, M. A. Howard, and

H. Kawasaki, "Intracranial EEG for seizure focus localization: evolving techniques, outcomes, complications, and utility of combining surface and depth electrodes," *J. Neurosurg.*, pp. 1–13, May 2018.

[7] S. Ramgopal, S. Thome-Souza, M. Jackson, N. E. Kadish, I. S. FernÃ¡ndez, J. Klehm, W. Bosl, C. Reinsberger, S. Schachter, and T. Loddenkemper, "Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy," *Epilepsy & Behavior*, vol. 37, pp. 291 – 307, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1525505014002297

[8] A. J. B. A. M., "Movement-based seizure detection," *Epilepsia*, vol. 59, no. S1, pp. 30–35, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/epi.14053

[9] T. D. Cooman, C. Varon, A. V. de Vel, K. Jansen, B. Ceulemans, L. Lagae, and S. V. Huffel, "Adaptive nocturnal seizure detection using heart rate and low-complexity novelty detection," *Seizure*, vol. 59, pp. 48 – 53, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1059131118301468

[10] T. N. Alotaiby, S. A. Alshebeili, T. Alshawi, I. Ahmad, and F. E. Abd El-Samie, "Eeg seizure detection and prediction algorithms: a survey," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 183, Dec 2014. [Online]. Available: https://doi.org/10.1186/1687-6180-2014-183

[11] A. K. Jaiswal and H. Banka, "Local pattern transformation based feature extraction techniques for classification of epileptic eeg signals," *Biomedical Signal Processing and Control*, vol. 34, pp. 81 – 92, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S174680941730006X

[12] R. Hussein, H. Palangi, R. Ward, and Z. J. Wang, "Epileptic seizure detection: A deep learning approach," *arXiv preprint arXiv:1803.09848*, 2018.

[13] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, S. Ippolito, and O. Kavehei, "Convolutional neural networks for seizure prediction using intracranial and scalp electroencephalogram," *Neural Networks*, vol. 105, pp. 104 – 111, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608018301485

[14] H. Shiao, V. Cherkassky, J. Lee, B. Veber, E. E. Patterson, B. H. Brinkmann, and G. A. Worrell, "Svm-based system for prediction of epileptic seizures from ieeg signal," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 5, pp. 1011–1022, May 2017.

[15] W. Zhou, Y. Liu, Q. Yuan, and X. Li, "Epileptic seizure detection using lacunarity and bayesian linear discriminant analysis in intracranial eeg," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 12, pp. 3375–3381, Dec 2013.

[16] W. Stacey, M. L. V. Quyen, F. Mormann, and A. Schulze-Bonhage, "What is the present-day eeg evidence for a preictal state?" *Epilepsy Research*, vol. 97, no. 3, pp. 243 – 251, 2011, special Issue on Epilepsy Research UK Workshop 2010 on "Preictal Phenomena". [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920121111002154

[17] C. C. Jouny, P. J. Franaszczuk, and G. K. Bergey, "Characterization of epileptic seizure dynamics using Gabor atom density," *Clin Neurophysiol*, vol. 114, no. 3, pp. 426–437, Mar 2003.

[18] C. Geier, S. Bialonski, C. E. Elger, and K. Lehnertz, "How important is the seizure onset zone for seizure dynamics?" *Seizure*, vol. 25, pp. 160–166, Feb 2015.

[19] N. M. Wetjen, W. R. Marsh, F. B. Meyer, G. D. Cascino, E. So, J. W. Britton, S. M. Stead, and G. A. Worrell, "Intracranial electroencephalography seizure onset patterns and surgical outcomes in nonlesional extratemporal epilepsy," *J. Neurosurg.*, vol. 110, no. 6, pp. 1147–1152, Jun 2009.

[20] J. S. Naftulin, O. J. Ahmed, G. Piantoni, J. B. Eichenlaub, L. E. Martinet, M. A. Kramer, and S. S. Cash, "Ictal and preictal power changes outside of the seizure focus correlate with seizure generalization," *Epilepsia*, vol. 59, no. 7, pp. 1398–1409, Jul 2018.

[21] C. Harden, T. Tomson, D. Gloss, J. Buchhalter, J. H. Cross, E. Donner, J. A. French, A. Gil-Nagel, D. C. Hesdorffer, W. H. Smithson, M. C. Spitz, T. S. Walczak, J. W. Sander, and P. Ryvlin, "Practice guideline summary: Sudden unexpected death in epilepsy incidence rates and risk factors: Report of the Guideline Development, Dissemination, and Implementation Subcommittee of the American Academy of Neurology and the American Epilepsy Society," *Neurology*, vol. 88, no. 17, pp. 1674–1680, Apr 2017.

[22] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," *Phys. Rev. E*, vol. 64, p. 061907, Nov 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.64.061907

[23] S. Kaspar, G. Heidemarie, G. Marc, and R. Christian, "On seeing the trees and the forest: Singlesignal and multisignal analysis of periictal intracranial eeg," *Epilepsia*, vol. 53, no. 9, pp. 1658–1668, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1528-1167.2012.03588.x

[24] K. Schindler, H. Gast, L. Stieglitz, A. Stibal, M. Hauf, R. Wiest, L. Mariani, and C. Rummel, "Forbidden ordinal patterns of periictal intracranial eeg indicate deterministic dynamics in human epileptic seizures," *Epilepsia*, vol. 52, no. 10, pp. 1771–1780, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1528-1167.2011.03202.x

[25] Y. Kaya, M. Uyar, R. Tekin, and S. Yildirim, "1d-local binary

pattern based feature extraction for classification of epileptic eeg signals," *Applied Mathematics and Computation*, vol. 243, pp. 209 – 219, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300314008285

[26] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009. [Online]. Available: http://dx.doi.org/10.1007/s12559-009-9009-8

[27] R. W. Gayler, "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," in *Proceedings of the Joint International Conference on Cognitive Science. ICCS/ASCS*, 2003, pp. 133–138.

[28] T. Plate, *Holographic Reduced Representations*. CLSI Publications, 2003.

[29] A. Rahimi, S. Datta, D. Kleyko, E. P. Frady, B. Olshausen, P. Kanerva, and J. M. Rabaey, "High-dimensional computing as a nanoscalable paradigm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 9, pp. 2508–2521, Sept 2017.

[30] H. Li, T. F. Wu, A. Rahimi, K. S. Li, M. Rusch, C. H. Lin, J. L. Hsu, M. M. Sabry, S. B. Eryilmaz, J. Sohn, W. C. Chiu, M. C. Chen, T. T. Wu, J. M. Shieh, W. K. Yeh, J. M. Rabaey, S. Mitra, and H. S. P. Wong, "Hyperdimensional computing with 3D VRRAM in-memory kernels: Device-architecture co-design for energy-efficient, error-resilient language recognition," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec 2016, pp. 16.1.1–16.1.4.

[31] T. Wu, P.-C. Huang, A. Rahimi, H. Li, M. Shulaker, J. M. Rabaey, H.-S. Wong, and S. Mitra, "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study," in *IEEE International Solid-State Circuits Conference, ISSCC*, In press 2018.

[32] A. Rahimi, S. Benatti, P. Kanerva, L. Benini, and J. M. Rabaey, "Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition," in *IEEE International Conference on Rebooting Computing*, October 2016.

[33] A. Moin, A. Zhou, A. Rahimi, S. Benatti, A. Menon, S. Tamakloe, J. Ting, N. Yamamoto, Y. Khan, F. Burghardt, L. Benini, A. C. Arias, and J. M. Rabaey, "An EMG gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier," in *IEEE International Symposium on Circuits and Systems, ISCAS*, In press 2018.

[34] D. Kleyko, A. Rahimi, D. Rachkovskij, E. Osipov, P. Kanerva, and J. M. Rabaey, "Binary hyperdimensional computing: Trade-offs in choice of density and mapping characteristics," in *IEEE Transactions on Neural Networks and Learning Systems, TNNLS*, In press 2018.

[35] A. Rahimi, P. Kanerva, J. del R Millán, and J. M. Rabaey, "Hyperdimensional computing for noninvasive brain–computer interfaces: Blind and one-shot classification of EEG error-related potentials," *10th ACM/EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 2017.

[36] A. Rahimi, A. Tchouprina, P. Kanerva, J. d. R. Millán, and J. M. Rabaey, "Hyperdimensional computing for blind and one-shot classification of EEG error-related potentials," *Mobile Networks and Applications*, Oct 2017. [Online]. Available: https://doi.org/10.1007/s11036-017-0942-6

[37] M. Schmuck, L. Benini, and A. Rahimi, "Hardware Optimizations of Dense Binary Hyperdimensional Computing: Rematerialization of Hypervectors, Binarized Bundling, and Combinational Associative Memory," *ArXiv e-prints*, Jul. 2018.

[38] M. Laiho, J. H. Poikonen, P. Kanerva, and E. Lehtonen, "High-dimensional computing with sparse vectors," in *Biomedical Circuits*

and Systems Conference (BioCAS), 2015 IEEE, Oct 2015, pp. 1–4.

[39] P. Kanerva, "Binary spatter-coding of ordered k-tuples," in *ICANN'96, Proceedings of the International Conference on Artificial Neural Networks*, ser. Lecture Notes in Computer Science, , Ed., vol. 1112. Springer, 1996, pp. 869–873.

[40] ——, *Sparse Distributed Memory*. Cambridge, MA, USA: The MIT Press, 1988.

[41] M. A. Kelly, D. Blostein, and D. J. K. Mewhort, "Encoding structure in holographic reduced representations." *Canadian Journal of Experimental Psychology*, vol. 67, no. 2, pp. 79–93, 2013.

[42] K. Schindler, C. Rummel, R. G. Andrzejak, M. Goodfellow, F. Zubler, E. Abela, R. Wiest, C. Pollo, A. Steimer, and H. Gast, "Ictal time-irreversible intracranial eeg signals as markers of the epileptogenic zone," *Clinical Neurophysiology*, vol. 127, no. 9, pp. 3051 – 3058, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138824571630459X

[43] L. Wang, X. Long, J. B. Arends, and R. M. Aarts, "Eeg analysis of seizure patterns using visibility graphs for detection of generalized seizures," *Journal of Neuroscience Methods*, vol. 290, pp. 85 – 94, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027017302510

[44] L. Lacasa, A. Nuñez, É. Roldán, J. M. R. Parrondo, and B. Luque, "Time series irreversibility: a visibility graph approach," *The European Physical Journal B*, vol. 85, no. 6, p. 217, Jun 2012. [Online]. Available: https://doi.org/10.1140/epjb/e2012-20809-8

[45] C. S. Daw, C. E. A. Finney, and E. R. Tracy, "A review of symbolic analysis of experimental data," *Review of Scientific Instruments*, vol. 74, no. 2, pp. 915–930, 2003. [Online]. Available: https://doi.org/10.1063/1.1531823

[46] B. Jun and D. Kim, "Robust face detection using local gradient patterns and evidence accumulation," *Pattern Recognition*, vol. 45, no. 9, pp. 3304 – 3316, 2012, best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320312001094

[47] K. Gröchenig, *The Short-Time Fourier Transform*. Boston, MA: Birkhäuser Boston, 2001, pp. 37–58. [Online]. Available: https://doi.org/10.1007/978-1-4612-0003-1_4

[48] F. Rosenow and H. Luders, "Presurgical evaluation of epilepsy," *Brain*, vol. 124, no. 9, pp. 1683–1700, 2001. [Online]. Available: http://dx.doi.org/10.1093/brain/124.9.1683

[49] K. Weaver, V. Morales, S. Dunn, K. Godde, and P. Weaver, *An Introduction to Statistical Analysis in Research: With Applications in the Biological and Life Sciences*. Whiley, 2017.

[50] Y. Liu, W. Zhou, Q. Yuan, and S. Chen, "Automatic seizure detection using wavelet transform and svm in long-term intracranial eeg," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 20, no. 6, pp. 749–755, 2012.

[51] D. Geng, W. Zhou, Y. Zhang, and S. Geng, "Epileptic seizure detection based on improved wavelet neural networks in long-term intracranial eeg," *Biocybernetics and Biomedical Engineering*, vol. 36, no. 2, pp. 375 – 384, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0208521616300250

[52] E. Alickovic, J. Kevric, and A. Subasi, "Performance evaluation of empirical mode decomposition, discrete wavelet transform, and wavelet packed decomposition for automated epileptic seizure detection and prediction," *Biomedical Signal Processing and Control*, vol. 39, pp. 94 – 102, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1746809417301544

[53] F. Rosenow and H. Luders, "Presurgical evaluation of epilepsy,"

Brain, vol. 124, no. 9, pp. 1683–1700, 2001. [Online]. Available: http://dx.doi.org/10.1093/brain/124.9.1683

[54] P. Swami, T. K. Gandhi, B. K. Panigrahi, M. Tripathi, and S. Anand, "A novel robust diagnostic model to detect seizures in electroencephalography," *Expert Systems with Applications*, vol. 56, pp. 116–130, 2016.

[55] D. Hernández, L. Trujillo, E. Z-Flores, O. Villanueva, and O. Romo-Fewell, "Detecting epilepsy in eeg signals using time, frequency and time-frequency domain features," in *Computer Science and EngineeringTheory and Applications*. Springer, 2018, pp. 167–182.

[56] P. Branco, L. Torgo, and R. P. Ribeiro, "A survey of predictive modeling on imbalanced domains," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 31, 2016.

[57] H.-T. Shiao, V. Cherkassky, J. Lee, B. Veber, E. E. Patterson, B. H. Brinkmann, and G. A. Worrell, "Svm-based system for prediction of epileptic seizures from ieeg signal." *IEEE Trans. Biomed. Engineering*, vol. 64, no. 5, pp. 1011–1022, 2017.

[58] A. H. Ansari, V. Matic, M. De Vos, G. Naulaers, P. Cherian, and S. Van Huffel, "Improvement of an automated neonatal seizure detector using a post-processing technique," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE, 2015, pp. 5859–5862.

[59] V. Cherkassky, B. Veber, J. Lee, H. T. Shiao, N. Patterson, G. A. Worrell, and B. H. Brinkmann, "Reliable seizure prediction from eeg data," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[60] H. Martin, A. DirkMatthias, and S. Andreas, "Latencies from intracranial seizure onset to ictal tachycardia: A comparison to surface eeg patterns and other clinical signs," *Epilepsia*, vol. 56, no. 10, pp. 1639–1647, 2015. [Online]. Available: https:

//onlinelibrary.wiley.com/doi/abs/10.1111/epi.13117

[61] P. Afra, C. C. Jouny, and G. K. Bergey, "Duration of complex partial seizures: an intracranial EEG study," *Epilepsia*, vol. 49, no. 4, pp. 677–684, Apr 2008.

[62] G. K. Bergey, "Neurostimulation in the treatment of epilepsy," *Exp. Neurol.*, vol. 244, pp. 87–95, Jun 2013.

[63] S. Wolfram, "Random sequence generation by cellular automata," *Advances in Applied Mathematics*, vol. 7, no. 2, pp. 123 – 169, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/019688588690028X

[64] O.-Y. Kwon and S.-P. Park, "Depression and anxiety in people with epilepsy," *Journal of Clinical Neurology*, vol. 10, pp. 175 – 188, 2014. [Online]. Available: http://synapse.koreamed.org/DOIx.php?id=10.3988

[65] K. Schindler, H. Leung, C. E. Elger, and K. Lehnertz, "Assessing seizure dynamics by analysing the correlation structure of multichannel intracranial eeg," *Brain*, vol. 130, no. 1, pp. 65–77, 2007. [Online]. Available: http://dx.doi.org/10.1093/brain/awl304