VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Le Trung Thanh

EEG EPILEPTIC SPIKE DETECTION USING DEEP BELIEF NETWORKS

Major: Electronics and Communications

Hanoi - 2016

VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Le Trung Thanh

EEG EPILEPTIC SPIKE DETECTION USING DEEP BELIEF NETWORKS

Major: Electronics and Communications

Supervisor: Assoc. Prof. Dr. Nguyen Linh Trung
Co-Supervisor: Dr. Le Vu Ha

Hanoi - 2016

AUTHORSHIP

" I hereby declare that the work contained in this thesis is of my own and has not been previously submitted for a degree or diploma at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no materials previously or written by another person except where due reference or acknowledgement is made."

Signature: ....

SUPERVISOR'S APPROVAL

" I hereby approve that the thesis in its current form is ready for committee examination as a requirement for the Bachelor of Electronics and Communications degree at the University of Engineering and Technology. "

Signature: ....

ACKNOWLEDGEMENT

This dissertation would not have been possible without the guidance and the help of several individuals who contributed and extended their valuable assistance in the preparation and completion of this study.

I am deeply thankful to my parents, who have been sacrificing their whole life for me and always supporting me throughout my education process.

I would like to express my sincere gratitude to my supervisors, professors Nguyen Linh Trung and Le Vu Ha, who introduced me to the interesting research problem of EEG signal analysis that combines signal processing in electrical engineering and machine learning in computer science. I would like to thank professor Tran Quoc Long for his valuable comments on deep learning tools. The work in this thesis is part of the Project on "Automatic detection of epileptic spikes based on deep learning", conducted by professor Tran Quoc Long.

I am grateful to all members of the Signals and Systems Laboratory, especially to my class-mate Dinh Van Viet who helped explain me the basics of EEG, my tutors Nguyen Thi Anh Dao and Le Thanh Xuyen for all their enthusiastic guidance and encouragement during the study and preparation for my thesis.

Finally, I would like to express my great appreciation to all professors of the Faculty of Electronics and Telecommunications for their kind teaching during the four years of my study.

Le Trung Thanh.

ABSTRACT

In the clinical diagnosis of epilepsy using electroencephalogram (EEG) data, accurate automatic detection (also called classification) of epileptic spikes is useful and highly meaningful. In this thesis, we apply Deep Belief Network (DBN), that is one of deep learning models, to replace Artificial Neural Network (ANN) stage in a recently proposed multi-stage system for epileptic spike detection. The proposed method is original as, to the best of our knowledge, deep learning has not been used for spike detection. It is also useful in practice because the promising quality evaluation of the spike detection system will be higher than 90%. More specifically, to construct the accurate detection model for non-spikes and spikes, a new set of features (wavelet scales from 3 to 9) was proposed that gives a good description of spikes. These features were then used as inputs to the DBN. A performance comparison between using the DBN and ANN was provided via numerical study by simulation. Accordingly, the sensitivity obtained by using the DBN and the ANN is 93.42% and 84.57%, respectively. These results indicate that it is possible to use deep learning models for epileptic spike detection with high accuracy.

Index Terms: Electroencephalogram (EEG), Artificial Neural Network (ANN), Deep Belief Network (DBN).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVATIONS

Abbreviation	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DBM	Deep Boltzmann Machine
EEG	Electroencephalogram
GMM	Gaussian Mixture Model
MLP	Multilayer Perceptron
MCMC	Marko Chain Monte Carlo
NN	Neural Network
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SVM	Support Vector Machine

Chapter 1

INTRODUCTION

1.1 Epilepsy and EEG

Epilepsy is a chronic disorder of the nervous system in the brain characterized by epileptic seizures which are due to abnormal, excessive discharges of nerve cells. The symptoms of epilepsy depend on the affected area of the brain, also called legion. Generally, people with epilepsy may have uncontrollable movements, loss of consciousness awareness, and temporary confusion. These may lead to other serious problems such as the higher proportion of traffic accidents and birth-related injuries. Although the causes of epilepsy are quite varied, we could divide them into three main groups: symptomatic (such as traumatic head injuries, stocks, brain tumors, brain infections), idiopathic (which involves genetic influence), and cryptogenic epilepsy (with unidentifiable causes) [1].

Statistics have been illustrating the serious situation that epilepsy may cause. According to recent data from Epilepsy Foundation [2], there are now approximately 65 million people diagnosed epilepsy and about 2.4 million people detected signs of the disease each year in the world. This makes epilepsy is the fourth most common neurological diseases globally. The number of new cases is between 30 and 50 per 100.000 people in the general population in high-income countries. In developing countries, the figures are twice as high as in the developed countries [3]. The figures are remarkable and can increase significantly in the future.

The diagnosis of epilepsy is typically based on observation of the seizure onset and the underlying cause. Electroencephalogram (EEG) is one of the most accessible tools supporting this observation. It refers to the recording of electrical activity of the brain by measuring voltage fluctuations resulting from ionic current flows within the neurons of the brain. The measurement is done by using sensors (electrodes) attached to the head, receiving electrical impulses of the brain and sending them to a computer. The electrical impulses in an EEG recording is normally characterized by wavy lines with peaks and valleys. From the EEG recordings, one can detect and classify the specified kinds of signals representing the body status. In particular, doctors usually inspect the EEG recordings on a computer screen and look for epileptic spikes, which are abnormal patterns of the brain electrical activity. This conventional manual process is not only very tedious and time-consuming, but also subjective since it depends on the knowledge and experience of the doctors. Furthermore, epilepsy may have similar signs in EEG data to other diseases, including psychogenic non-epileptic seizures (PNES), panic attacks and hyperventilation syndrome. These drawbacks encourage the implementation of a new fully or partially automated EEG support system for spike detection, which will be investigated in the next section.

1.2 Automatic epileptic spike detection

In the procedure of the diagnosis of epilepsy, automatic epileptic spike detection is important because it saves time and can provide much information. In last four decades, many methods have been proposed for automatic spike detection based on (1) comparison thresholds of true spikes with parameters of possible spikes, (2) filtering techniques or (3) spike detection system. Among them, a spike detection system often provides the best results since it typically allows to combine and cumulate the advantages of multiple algorithms. A spike detection system can be either single-stage or multi-stage. The latter is generally more efficient.

In the multi-stage systems, one of the most important steps is the application of machine learning models to learn and recognize different patterns of EEG data. Machine learning is a data mining method which enables the computer to learn the complex, hidden features in data without explicit programming. This is a powerful tool but can be the bottleneck of the system. In fact, the performance of existing systems has reached only 85% on average so far. This can be explained by the fact that the current learning models, such as Artificial Neural Network (ANN [4], K-means clustering [5], and Support Vector Machine [6]) are shallow, as discussed shortly, while the epileptic spikes usually have complicated features. This motivates us to look for more powerful models. Deep learning can be the solution.

1.3 Deep learning for spike detection

Before showing how deep learning can be the solution, we shall first argue the drawback of the afore learning models. Until recently, most of problems in data analytics and signal processing, and the problem of automatic epileptic spike detection in particular, have been using shallow architectures [7]. In general, these architectures are composed of just two layers at most with non-linear functions such as logistic regression, Support Vector Machine (SVM), Gaussian mixture model (GMM). The shallowness blocks those models to learn high-level abstractions, limits their ability of dealing with complex real applications like natural language and image processing. This raises the need of deep architectures and eventually deep learning algorithms.

In recent years, deep learning has been attracting a great attention in research community on machine learning. It is based on a set of algorithms of learning the multi levels of representation and abstraction. Deep architectures have been applied in many applications and achieved several successes in *Big Data*. Deep learning has surprised the community with some outstanding practical results in natural language processing, speech recognition, image processing. For example, in June 2013, a deep learning system by Google was able to process 10 million images from Youtube videos, which is almost twice as good as any former image processing methods at identifying objects. Recently, in March 2016, the deep learning-aided computer Alphago beat the 9-dan professional player Lee Sedol for the first time ever in a historical chess match. The potential of deep architecture models in analyzing recorded EEG signal is obvious. Indeed, deep learning has been exploited in some EEG research. For example, Convolutional Neuron Network (CNN) is applied for seizure prediction in EEG in [8]. In [9], Zheng *et al.* used Deep Belief Network (DBN) to investigate the critical frequency bands and channels for EEG-Based Emotion Recognition. Wulsin *et al.* used DBN to detect anomalies related to epilepsy in EEG recording [10]. The deep learning techniques is demonstrated by Plis *et al.* for neuron-imaging [11].

1.4 Aim and architecture of the thesis

In this thesis, we would like to use Deep Belief Network as a classifier to detect epileptic spikes in EEG signal. We would like to combine multi-step of preprocessing stage including small spike elimination, filtering, perceptrons; extract features derived from multi-scale wavelet transform of the EEG signal and then train DBN to distinguish between the spikes and non-spikes.

The thesis is organized as the following. In Chapter 2, we describe the background of the study, including the multi-stage system, feature extraction, and a brief survey of deep architectures as well as Deep Belief Network. We introduce our proposed method to improve this multi-stage system in 3 and show the experimental results in Chapter 4. Finally, Chapter 5 concludes the thesis with some notes and future works.

Chapter 2

BACKGROUND

2.1 A multi-stage system for epileptic spike detection

As mentioned in the previous chapter, the automatic epileptic spike detection system is important and normally more efficient than manual clinic. Generally, the systems for epileptic spikes can be divided into two main classes: single-stage and multi-stage. In the first class, the architecture systems are often very simple composed of one or two techniques, while many signal processing and machine learning methods build up multi-stage systems in order to obtain high performance. In this thesis, we inherit a recently proposed multistage system in [12] and introduce some new methods making use of deep learning to improve its performance.

2.1.1 System architecture

First, we would like to describe this multi-stage spike detection system, whose structure is shown in Fig 2.1, in the following. Assume that we have the EEG recording at hand and input it to the system. The system then process it in four main steps as follows.

1. *Preprocessing*: The system first detects all peaks in the EEG signal, then identifies and removes the small peaks, whose amplitude and duration are less than 20ms and 2μV, respectively. After that, for each remaining significant peaks, 6 spike features

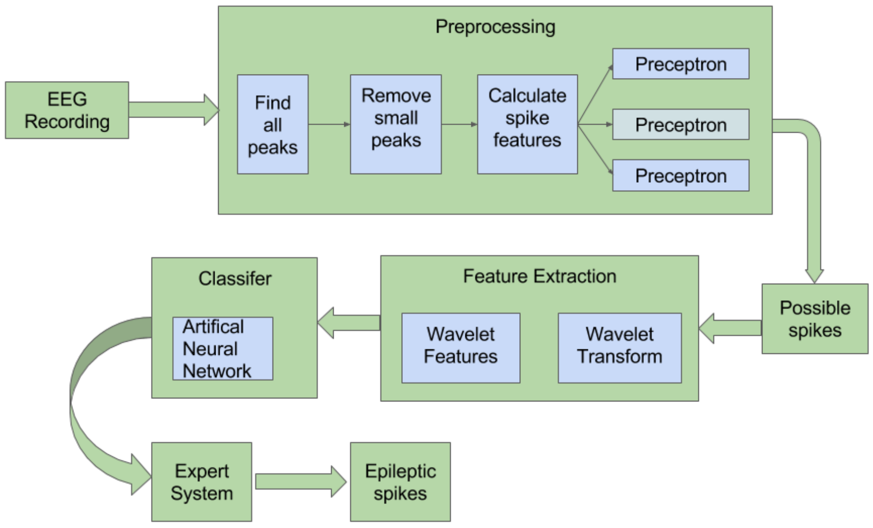


Figure 2.1: Multi-stage system for epileptic spike detection

characterizing durations, amplitudes and slopes are extracted including First Half Wave Amplitude (a1), First Half Wave Duration (d1), Second Half Wave Amplitude (a2), Second Half Wave Duration (d2), First Half Wave Slope (s1) and Second Half Wave Slope (s2) as shown in Fig 2.2. These features are then used as inputs of three different neural network perceptrons which divide the peaks into two groups: non-spike and possible spike.

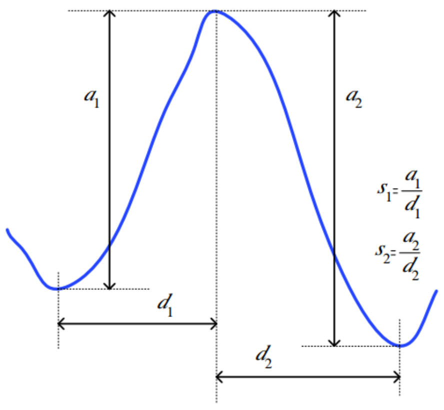


Figure 2.2: Six features of a spike.

2. *Feature Extraction*: The possible spikes are analyzed by continuous wavelet transform. The transformed signal corresponding to each scale are then used to calculate

7 wavelet features, as shown in Fig 2.4, to resemble an epileptic spike. According to [12], the 4^{th} to the 8^{th} wavelet scales were chosen for wavelet transform to obtain an input vector of 35 units for classification stage. We will clarify the feature extraction using wavelet transform in the next section and introduce a small modification in wavelet scale range to be suitable for a Deep Belief Network in Chapter 3.

3. *Classification*: Features extracted from wavelet transform are fed to a classification model producing a binary class label as its output. The spike score values are in the range $[0, 1]$. A peak is labeled a spike when its corresponding output is higher than a certain threshold defined by the model, and vice verse. This stage uses the Artificial Neural Network learning model which calculate probability of a test vector assigned to epileptic spike class, as discussed in Section 2.1.3. We will show how we apply Deep Belief Network model instead of using Artificial Neural Network to improve the system in Chapter 3.

4. *Expert System*: Finally, to ensure that the spike is correctly identified by the system, this system applies a simple rule which there are at least two spike in duration of 150-350 ms in order to eliminate the pseudo spikes which are located near an epileptic spike. In this study, we ignore this stage in order to focus on the direct outcome after the DBN classifier.

2.1.2 Feature extraction using wavelet transform

In this section, we present how wavelet transform can be the solution for the problem of feature extraction. The feature extraction step is highly necessary as it extracts the characterizing parameters from raw EEG data for classification. Some different methods have been proposed based on the parameters of spike in time domain or frequency domain such as eigen methods [13], a spike model with slow wave features in [14] to find a set of measurements/features characterizing the spikes. The measurements or features of spikes can be divided into some main groups, namely duration, amplitude, slope, and area.

The methods based on time-frequency distributions, like wavelet transform, would be efficient to obtain a sufficient set of information of spikes. In addition, wavelet transform is highly beneficial for non-stationary signal like EEG recordings, which can vary greatly from patient to patient. Wavelet transform has been successfully applied in recent works in EEG such as the work on spike detection and sorting in [15]. Therefore, using waveform features of wavelet coefficients as input of the classifier could be effective.

Wavelet transform (WT) uses inner products to measure the similarity between a signal and an analyzing function. In continuous wavelet transform (CWT), the analyzing function is called the mother function $\psi(.)$. It compares the signal to the shifted and compressed or stretched versions of a wavelet by changing time-shift b and scale a as follows:

$$C(a,b;f(t),\psi(t)) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty}\psi^*(\frac{t-b}{a})dt \tag{2.1}$$

where ψ^* is complex conjugate of the mother function. In this work, we use the Mexican hat wavelet as depicted in Fig 2.3. By comparing the shape of the Mexican hat wavelet mother function and the spike (in Fig. 2.2 and Fig. 2.4), we can see clearly the similarity between them. as the mother function because its waveform matches a true epileptic spike closely. The Mexican hat wavelet has the following formula

$$\psi(t) = \frac{2}{\sqrt{3}\pi^{1/4}}(1 - \frac{t^2}{\sigma^2}).e^{-t^2/2\sigma^2}. \tag{2.2}$$

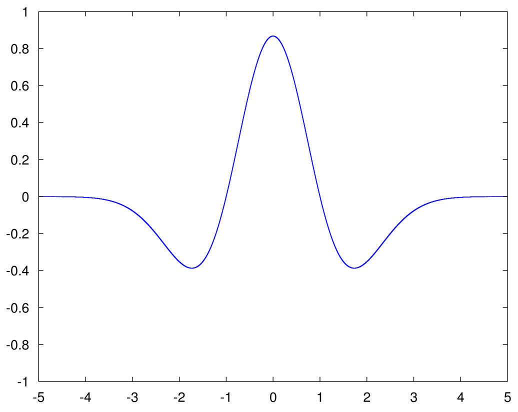


Figure 2.3: Mexican hat wavelet mother function.

Next, we analyze the wavelet features of a spike defined in [16] and depicted in Fig. 2.4.

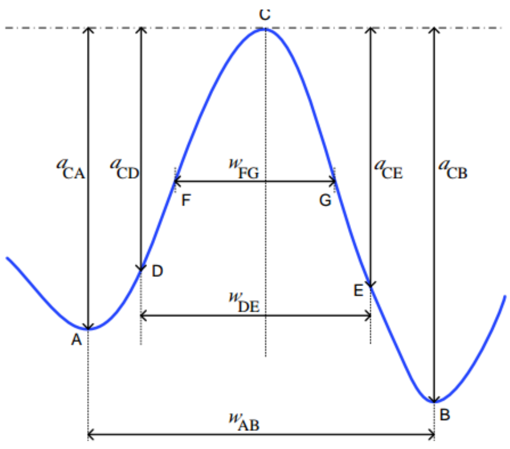


Figure 2.4: Wavelet feature parameters.

The features include:

- Relative amplitude 1 (RA_1): Voltage difference between the two successively peaks divided by the background amplitude: $RA_{1u} = \frac{a_{CA}}{w_T^n(p)}$, $RA_{1d} = \frac{a_{CB}}{w_T^n(p)}$ where $w_T^n(p)$ is the value of wavelet coefficients at peaks.

- Relative amplitude 2 (RA_2): Voltage difference between peak and turning point divided by the background amplitude: $RA_{2u} = \frac{a_{CD}}{w_T^n(p)}$, $RA_{2d} = \frac{a_{CE}}{w_T^n(p)}$.

- Duration 1 (W_1): Number of data points between the start point and the end point of the wave, equivalent to W_{AB} in Fig. 2.4.
- Duration 2 (W_2): Number of data points between the two turning points of the wave, equivalent to W_{DE} in Fig. 2.4.
- Duration 3 (W_3): Distance between the two *half-maximum points*, equivalent to W_{FG} in Fig. 2.4.
- Sharpness (SH): Changing rate of the slope at the peak point: $SH = \frac{sl(k+1) - sl(k-1)}{2}$ where $sl(k) = \frac{w^n(k+1) - w^n(k-1)}{2}$.
- Slope (SLP_1 and SLP_2): Slope of the lines connecting the peak and the two turning points: $SLP_1 = CN/DN$; $SLP_2 = CM/EM$.

The septuple features $(RA_1, RA_2, W_1, W_2, SH, SLP_1, SLP_2)$ are calculated for each wavelet scale which are combined into a input vector of the classification stage for epileptic spike detection.

2.1.3 Classification using ANN

The Artificial Neural Network (ANN) learning model is applied in the classification stage of the system in [12]. The architecture of ANN was constructed based on the neural structure of the brain. An ANN is modeled similarly to human brains biological neural networks in the performing of functions collectively and in parallel by the units, rather than by a clear delineation of subtasks to which individual units are assigned. Fig. 2.5 gives an visualization of ANN.

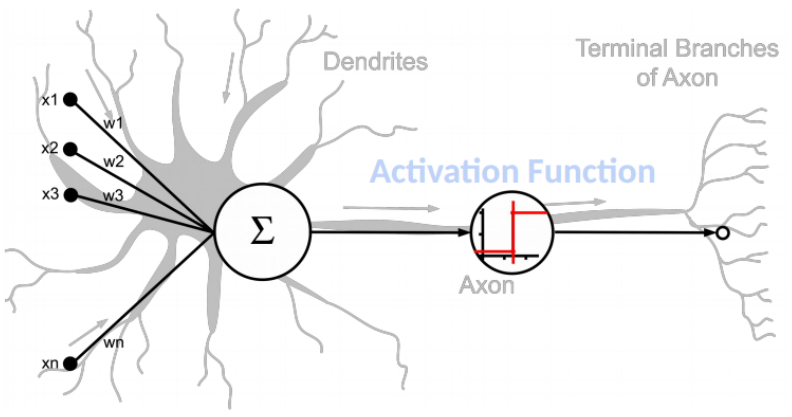


Figure 2.5: Artificial Neural Network.

Architecture of neural networks

ANN includes layers that are made up of a number of interconnected nodes which contain an activation function that defines the output of that node given an input or set of inputs, for instance some function shown in Fig. 2.6. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done via a system of weighted connections. The hidden layers then link to an output layer which gives the answer. ANN can be categorized into three types by its architectures, namely:

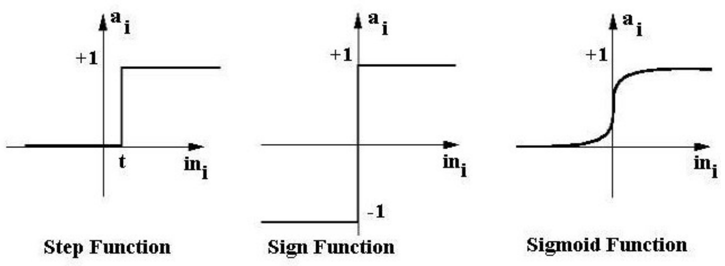


Figure 2.6: Activation functions of ANN.

- Feed-forward Networks: In feed-forward ANNs, signal (information) moves in only one direction from input to output. There are no loop or cycle in this network. This type of organization is also referred to as bottom-up or top-down.

- Feedback Networks: Feedback networks allow signal to move in both directions by introducing loops in the network. The networks are dynamic and very powerful in getting complicated features. Their architectures are also referred to interactive or recurrent. Feedback Networks Feed-forward Networks are depicted in Fig. 2.7.

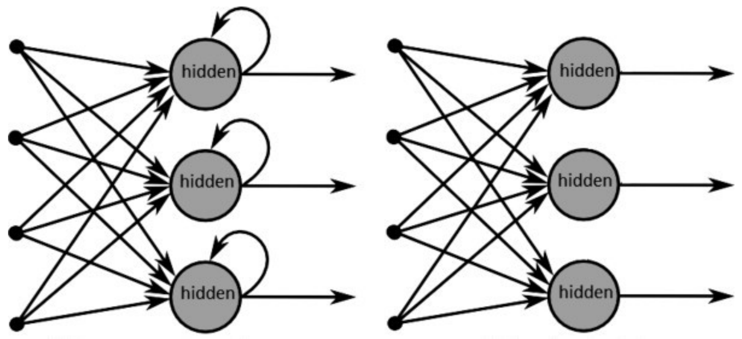


Figure 2.7: Feedback (on the left) and Feed-forward network (on the right).

- Perceptron: Perceptron, as shown in Fig. 2.8, is the simplest among three types of neural networks. The network takes a vector of feature activations converted from the raw input vector and learns the associated weights, in order to calculate a single scalar quantity for decision unit. If the value of the decision unit is above some threshold, then the input vector can be classified as the target class. Thus, the perceptron is an algorithm for supervised classification. The learning algorithm is simply adapting weights by minimizing the error between the desired output and the actual output. If the data is linearly separable, then the learning algorithm will converge. This simple network has many limitations, such as handcoded feature units are expensive and its single-layer structure makes it hard to learn a complex model.

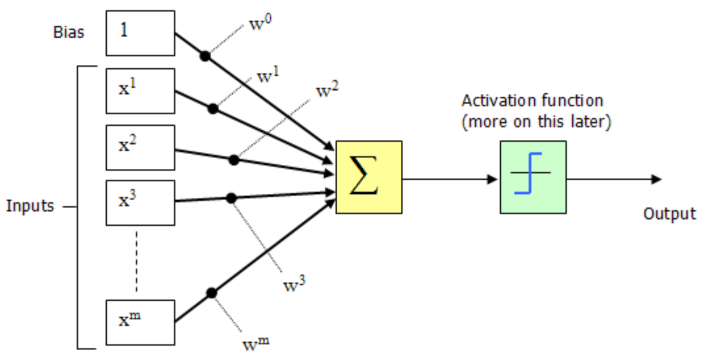


Figure 2.8: Perceptron Network.

Epileptic spike detection with ANN

The structure of an ANN for spike detection is illustrated in Fig 2.9 with three fully-connected layers: input, hidden and output [12]. The actual number of layers in the system is 41. Seven types of features (RA_1, RA_2, W_1, W_2, SH, SLP_1 and SLP_2) for five scales (from 4^{th} to 8^{th}) are input of the ANN.

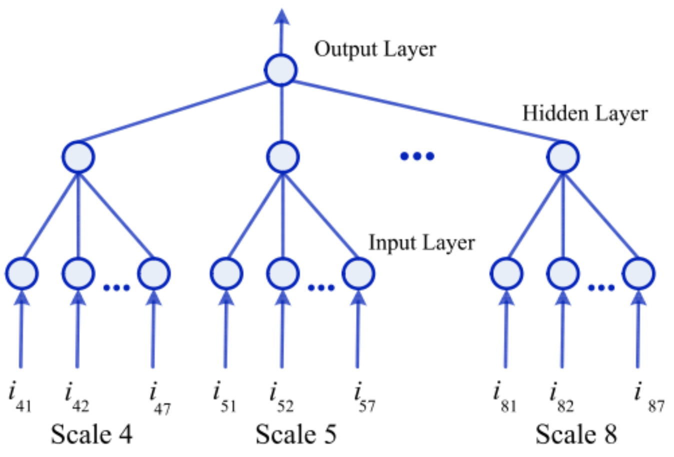


Figure 2.9: ANN for spike detection.

The training process is divided into 2 separate phases constituting a *twice-learning* scheme. First, ANN is trained by setting the desired output of value in the range of $[0, 1]$. The testing data is then passed through the ANN, yielding the output. The value corresponding to the true spike is closer to 1, and vice verse. Second, the output y are sorted for min-max values $\{y^s_{max}, y^s_{min}\}$ for spike group and $\{y^n_{max}, y^n_{min}\}$ for non-spike group and re-computed:

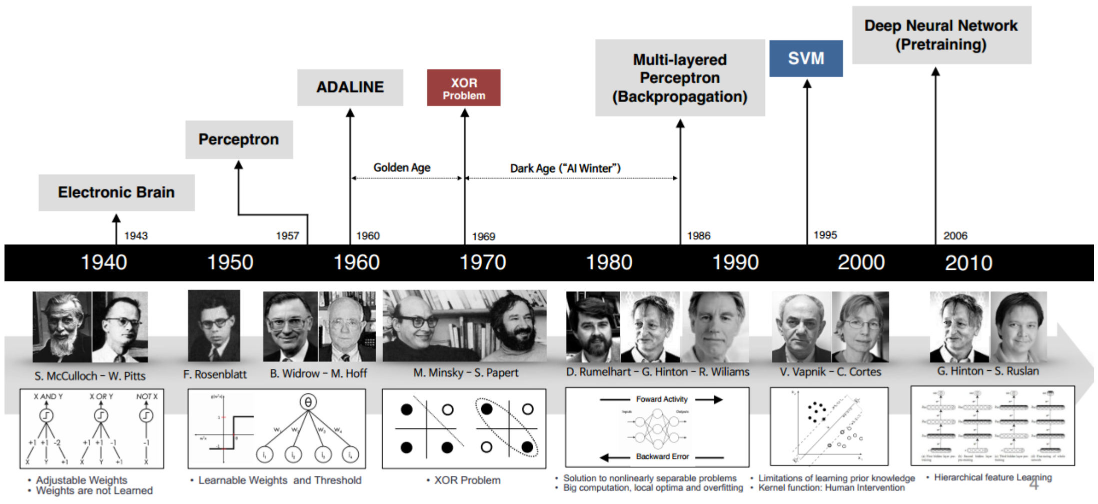
$$y^s = 0.45.\frac{y - y^s_{min}}{y^s_{max} - y^s_{min}} + 0.55 \tag{2.3}$$

$$y^n = 0.55.\frac{y - y^n_{min}}{y^n_{max} - y^n_{min}} \tag{2.4}$$

Then, the spike samples have the desired output in the range $(0.55, 1)$ and non-spike samples have the desired output in the range $(0, 0.45)$.

2.2 Deep Learning

Deep learning is a subfield of machine learning algorithms, referring to learning multiple levels of representation and abstraction. The concept of deep learning is related to artificial neural network research and paves the way for the third generation of neural networks (NN). A brief history of NN with an promising future thanks to deep learning can be consulted in Fig. 2.10. The first neural networks were developed in the 1950s [17], shortly after the beginning of Artificial Intelligence (AI) with promising expectations. The next generation of NN was an improvement, but had difficulties in recognizing complex patterns due to the limited number of neurons. This made it unused and faded away during the 1970s. In the mid-80s, Hinton and his co-workers made a *revolution* reviving the neural networks with deep models [7] by using multi neural layers. However, the methods still required a lot of human efforts as one had to assign labels for input before training. In 2006, Hinton had a breakthrough idea of a more effective way to learn neural layers by the so-called Greedy layer-wise training algorithm [18]. This idea ushered an exploding era of machine learning.



Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
ADALINE
XOR Problem
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Figure 2.10: Brief History of Neural Network.

The Greedy layer-wise training algorithm works as follows. The first layer learns basic features from data such as edges in images or the smallest units in sound. Once this layer has identified exactly the features, one continues to learn the second layer with more complex features such as a corner or a combination of voice audio. This process is repeated layer after layer until the system can recognize patterns or objects correctly.

2.2.1 Classes of deep learning networks

We could categorize deep leaning networks into into three main groups by their architectures and intended purposes (e.g. classification, recognition, and generation) [7] as follows.

Deep networks for unsupervised or generative learning

This group do not use the target class labels in the learning process and can be used to generate samples by sampling from the networks. Some well-known generative models in this group are Deep Boltzmann Machine (DBM), Deep Belief Network (DBN), Recurrent

Neural Network (RNN). They have the potential learning internal descriptions that are highly useful for recognizing speech pattern and modeling sequence data. Recursive generative models based on neural networks can found in [19] for human motion modeling, and in [20] for natural language.

Deep networks for supervised learning

Supervised learning infers a function from labeled training data. A RNN is a typical model in this class and can be used for discrimination. Indeed, a RNN has memory for remembering information calculated previously. The discriminative RNN is applied a long time ago and had some limit [7]. Consequently, some latter versions of RNN have been developed, such as Bidirectional RNN, Long short-term memory networks (LSTM) to surpass the disadvantages of RNN.

Another famous type of discriminative deep models is Convolutional Neural Network (CNN) in which each stage are composed of a convolutional layer and a pooling layer. The two layers share many weights. The pooling layer subsamples the output of the convolutional layer and reduces the data rate from the layer below, so it can be said that CNN is *invariance*. Although CNN is not applied for complex pattern recognition due to such limited *invariance*, they are highly effective and commonly used in computer vision, image recognition, and speech recognition. Besides, there are some improved versions of CNN changed, for instance, Time-delay Neural Network (TDNN), Convolutional Deep Belief Network (C-DBN).

Hybrid deep networks

The term *hybrid* refers to the deep architectures composed of both generative and discriminative model. The main goal of hybrid deep networks is discrimination. For instance, DBN, as a generative model for unsupervised learning, can be converted to supervised learning model with the same network structure. The discriminative criteria are used to estimate the parameters of generative models. Some other examples of hybrid deep networks are RBM/DBN, SVM/DBN (Support Vector Machine/Deep Belief Network), C-DBN/D-CNN (Deep CNN/Convolutional DBN). This hybrid deep learning approach can be applied to not only speech recognition but also other information processing like speech information retrieval, speech understanding, text understanding and retrieval.

We apply Deep Belief Network for epileptic spikes detection. The details of Deep Belief Network will be discussed in the following subsection.

2.2.2 Deep Belief Network

Deep Belief Network (DBN) was proposed by Hinton *et al.* in 2006 in [18] in order to overcome the disadvantages of former neural networks, namely Belief Network and Restricted Boltzmann Machine. We will describes these two neural networks first, and then present how DBN evolves from them.

Belief Network

Belief network is composed of 2-layers of stochastic binary units with weighted connection. The stochastic binary units in belief networks have a state of 0 or 1 and the probability of becoming 1 is determined by a bias and weighted input from other units:

$$p_i \equiv p(i=1) = \frac{1}{1+\exp(-\sum_j s_j w_{i,j})} \tag{2.5}$$

Fig. 2.11 provides an image of the belief network.

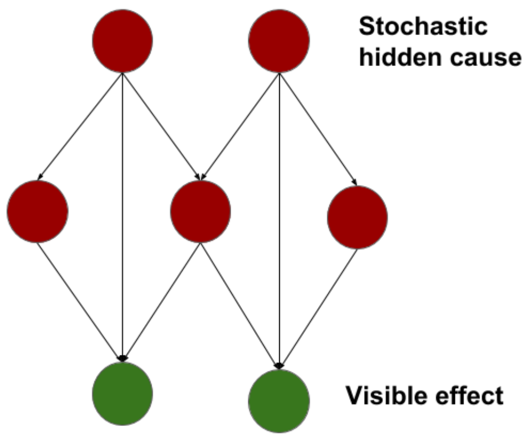


Figure 2.11: Belief Network.

The purpose of Belief Network are:

Inference: infer the state of unobserved variables;

Learning: adjust the interaction between variables to more likely generate the observed data.

The disadvantage of learning weights in belief networks is that in the case of conditional dependent, i.e. two independent hidden units become dependent and both influence, it is difficult to calculate the posterior distribution. Moreover, if belief networks have multi layers, then the posterior distribution depends on the prior and likelihood of upper hidden layers and there are numerous ways of possible configurations of these layers. Hinton *et al.* proved that learning one layer at a time and restricting the connectivity of stochastic binary units could obtain an approximate result [21, 22]. This is called Contrastive

Divergence algorithms and will be explained in the next subsection.

Restricted Boltzmann Machine

Restricted Boltzmann machine (RBM) is a special type of Markov random field, composed of two layers: *visible layer* where the state is observed, and *hidden layer* where the features are detected. There may be a symmetric connection between them while there is no connection between units within a layer [23]. The structure of a RBM is depicted in Fig. 2.12.

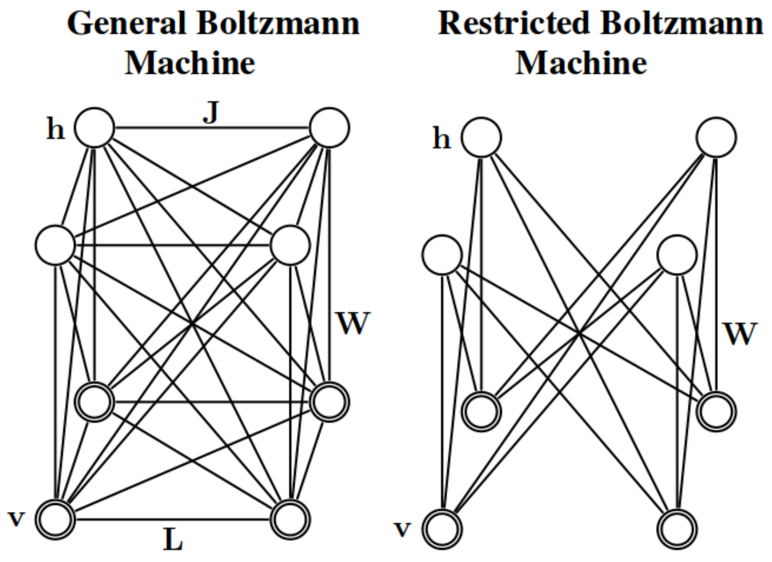


Figure 2.12: Restricted Bolzmann machine as compared to general Bolzmann machine.

In the configuration, the probability distribution over hidden and visible units are defined in terms of energy function:

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})} \tag{2.6}$$

where E is the *energy function*, defined as

$$E = -\sum v_i h_i W_{i,j} - \sum a_i v_i - \sum b_j h_j \tag{2.7}$$

and the Z is the *partition function*, given by summing all visible and hidden states:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{2.8}$$

The probability that the network assigns to a training input the visible vector v is

$$P(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{2.9}$$

This probability can be increased by adjusting the weights and biases to lower the energy of that input and to raise the energy of other visible vectors. In particular, when the energy of the visible vector of interest is low, it makes a big contribution to the partition function. Therefore, the selection of the parameter value can be found by solving this optimization problem:

$$\max_{w_{ij},a_i,b_j} \frac{1}{N}\sum_{l=1}^{N} \log \sum_{\mathbf{h}} P(\mathbf{v}^{(l)},\mathbf{h}^{(l)}) \tag{2.10}$$

where N is the number of training data. Taking derivative of the log probability (log likelihood) of training data, we have that

$$\frac{\partial \log P(v)}{\partial w_{i,j}} = \langle v_i h_j\rangle_0 - \langle v_i h_j\rangle_\infty. \tag{2.11}$$

This leads to a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data

$$\Delta W_{i,j} = \varepsilon.(\langle v_i h_j\rangle_0 - \langle v_i h_j\rangle_\infty) \tag{2.12}$$

$$\Delta a_i = \varepsilon.(\langle v_i\rangle_0 - \langle v_j\rangle_\infty) \tag{2.13}$$

$$\Delta b_j = \varepsilon.(\langle h_i\rangle_0 - \langle h_j\rangle_\infty) \tag{2.14}$$

where ε is the learning rate.

The goal of training is to generate the layer units that can be similar to observed data. Indeed, there are no direct connections between hidden units in an RBM, so it is easy to calculate $E_{data} \sim (\langle v_i h_j \rangle_0)$:

$$p(h_j = 1|v) = \frac{1}{1 + \exp(-b_j - \sum_i v_i W_{i,j})}. \tag{2.15}$$

It is also very easy to get an unbiased sample of the state of a visible unit given a hidden vector because there are no connections between units in visible layer:

$$p(v_i = 1|h) = \frac{1}{1 + \exp(-a_i - \sum_j h_j W_{i,j})}. \tag{2.16}$$

Obtaining the $E_{model}[v_i h_j] \sim \langle v_i h_j \rangle_\infty$, however, is much more difficult. It can be done by starting at any random state of the visible units and performing alternative Gibbs sampling for a very long time, as described in the MCMC algorithm [24] (Fig. 2.13). Furthermore, the Contrastive Divergence (CD) algorithm [21, 22] can be used to fasten the learning for an RBM. The general idea is to update all the hidden units in parallel starting with visible units, reconstruct visible units from the hidden units, and finally update the hidden units again. The intuition behind this is that after a few iterations the data will have moved from the target distribution (i.e. that of the training data) towards the proposed distribution, and so give an idea in which direction the proposed distribution should move to better model the training data. Empirically, Hinton has found that even 1 cycle of MCMC is sufficient for the algorithm to converge to the acceptable answer. The learning rule is

$$\Delta W_{i,j} = \varepsilon.(\langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_1). \tag{2.17}$$

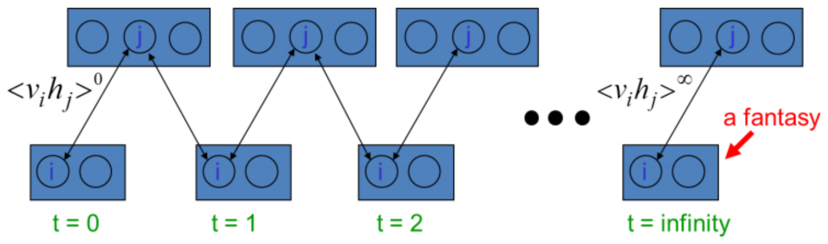


Figure 2.13: MCMC algorithm.

The CD algorithm with 1 cycle (CD_1) is summarized as follows:

- Initialize $\mathbf{v}_0$ at data;
- Sample $\mathbf{h}_0 := \mathbf{p}(\mathbf{h}|\mathbf{v}_0)$;
- Sample $\mathbf{v}_1 := \mathbf{p}(\mathbf{v}|\mathbf{h}_0)$;
- Sample $\mathbf{h}_1 := \mathbf{p}(\mathbf{h}|\mathbf{v}_1)$.

We have described the RBM. DBN can be viewed as a composition of RBMs, as explained in the next subsection. The idea is to allow multiple RBMs perform in the sequence to receive different representations of the data. After a RBM has been learned, the output of its hidden units can be used as the training data input for a higher RBM. Finally, a discriminative RBM is used as the last layer of the classifier.

Deep Belief Networks

DBN is a hybrid model composed of multi-layers obtained by stacking several RBMs on top of each other. Fig. 2.14 present the multi-layer structure of DBN. The hidden layer of the RBM at layer i becomes the input of the RBM at layer $i+1$. The layer-wise training process of DBN is shown in Fig 2.15 [25].

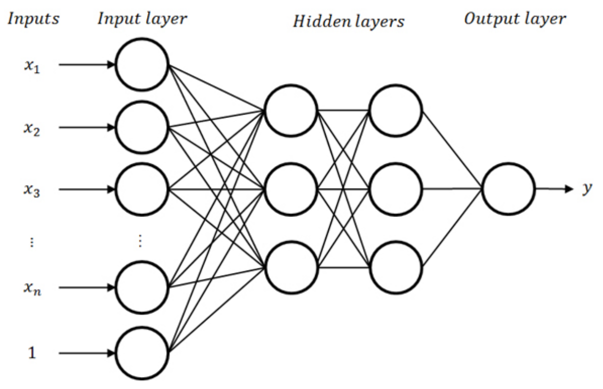


Figure 2.14: Deep Belief Network structure.

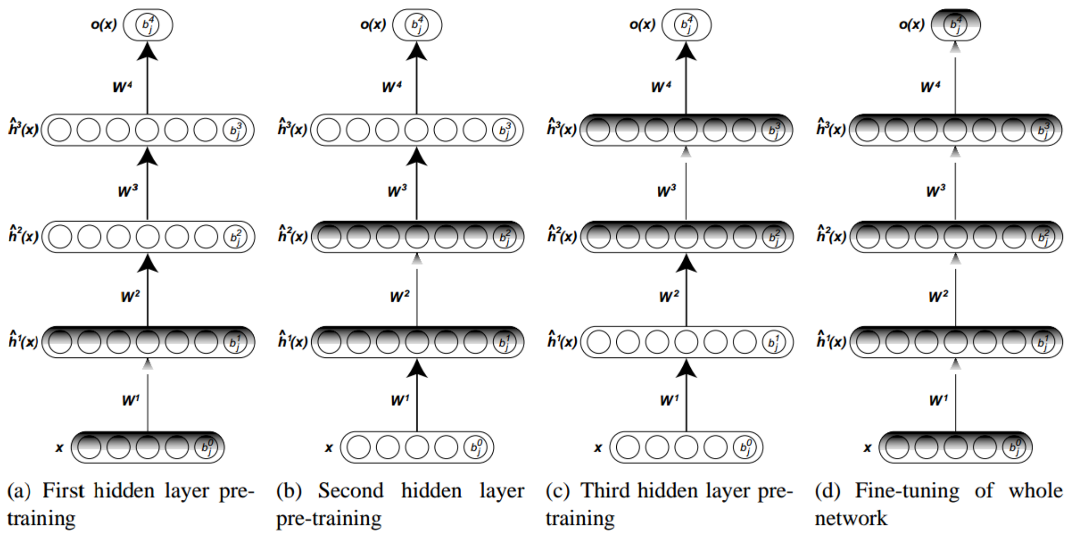

o(x)
W^4
$\hat{h}^3(x)$
W^3
$\hat{h}^2(x)$
W^2
$\hat{h}^1(x)$
W^1
x
(a) First hidden layer pre-training
(b) Second hidden layer pre-training
(c) Third hidden layer pre-training
(d) Fine-tuning of whole network

Figure 2.15: Training process of Deep Belief Network.

In particular, the first step of the training process is to learn a layer of features from the visible units, using Contrastive Divergence algorithm [22]. Then, the next step is to treat the activations of previously trained features as visible units and learn features of features in a second hidden layer. Finally, the whole DBN is trained when the learning for the final hidden layer is achieved. When used for classification, the DBN is treated as an MLP, by adding a logistic regression layer or using discriminative RBMs on top.

Chapter 3

THE PROPOSED METHOD

We improve the multi-stage system proposed in [12] by (i) modifying the feature extraction stage with 32 steps in scale range [3,9], and (ii) replacing the ANN in the classification stage with DBN. The structure of the modified system is shown in Fig. 3.1.

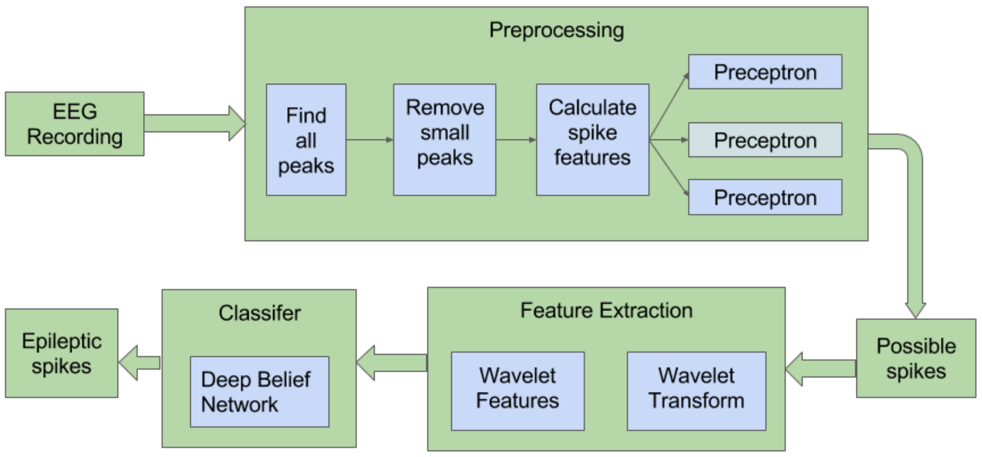
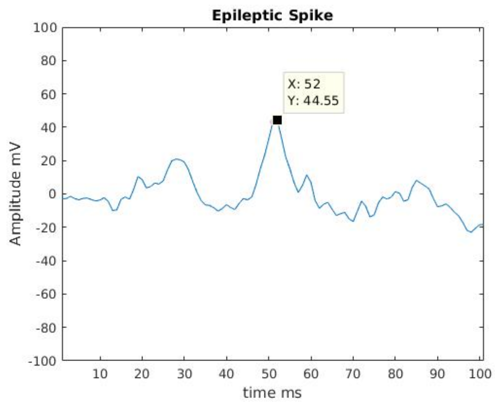


Figure 3.1: Multi-stage automatic spike detection system using DBN.

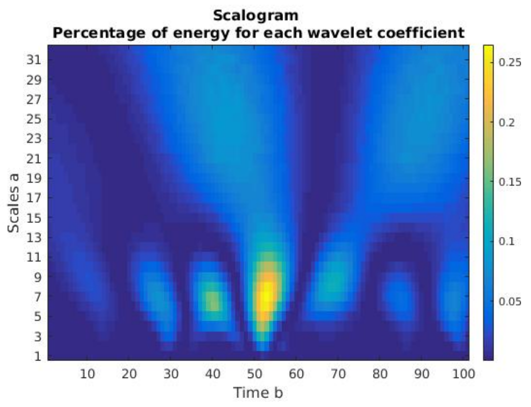
3.1 Feature extraction with 32 steps in the scale range [3,9]

As discussed in the previous chapter, wavelet transform is used in feature extraction stage due to its well-known efficiency in non-stationary signal analysis. Wavelet transform provides a different view of signal and visualizes the signal features. In particular, wavelet transform decomposes an original signal $x(t)$ into different signals in item of frequency and time by varying the wavelet scale a and shift b.

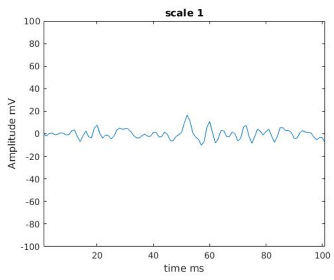
Wavelet features are obtained immediately from the waveform of the transformed signal, so the selection of wavelet scale is every important to balance or unbalance between the role of wavelet features for spike detection system using them as input. The wavelet scale is selected such that the corresponding transformed signal of an epileptic spike is likely to be waveform of the true spike, while wavelet transform of non-spike is disabled. In the multi-stage automatic epileptic spike detection system in [12], the authors choose the continuous wavelet transform (CWT) at 5 scales (from 4^{th} to 8^{th}). In this work, we change in the scale range to 7 scales from 3^{th} to 9^{th} to be suitable for our EEG dataset. This is motivated by the observation in the Fig 3.2 that wavelet coefficients of epileptic spikes are often dominant in the scale range $[3, 9]$ with a peak at scale 6 while the transformed signals of non-spikes may be eliminated, as shown in. There is a large *distance* between spike and non-spike in this range that would make the classification more efficient. In addition, by enlarging the scale range, we increase the dimension of vector input space and provide more information about input. This is compatible with the deep learning model, which can learn multi-level representation of data, that we use in the classification stage. However, to reduce the computational complexity, we only divide the scale range $[3, 9]$ into 32 steps to obtain 224 parameters of features. These parameters are then fed to the DBN classifier as discussed in the next section.



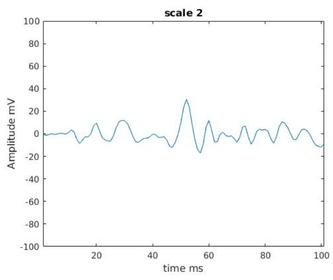
(a)



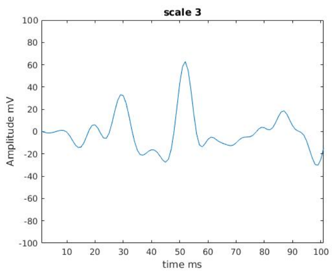
(b)



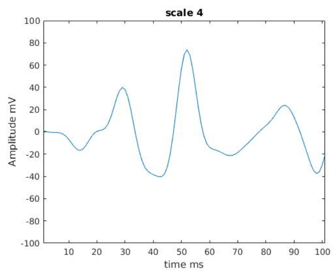
(c)



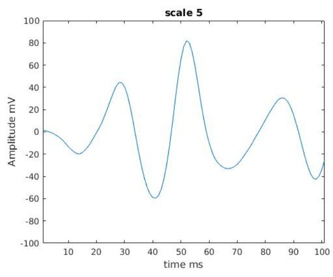
(d)



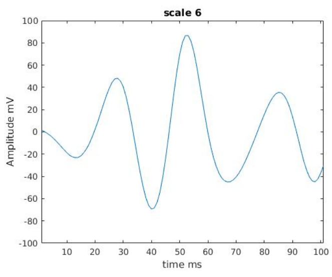
(e)



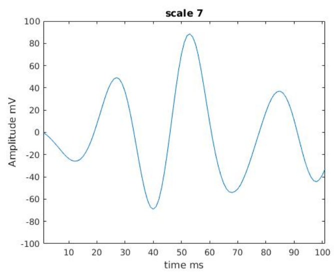
(f)



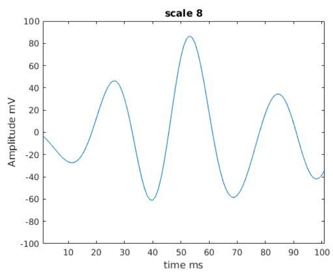
(g)



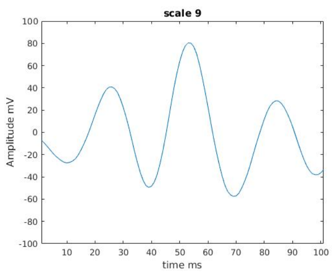
(h)



(i)



(j)



(k)

Figure 3.2: CWT of a epileptic spike with different scales.

3.2 Classification with the Deep Belief Network

In the previous chapter, we already explained the principles of RBM and DBN. In this section, we clarify our use of DBN in the multi-stage system for epileptic spike detection.

Structure of DBN for classification

Recall that DBN is a deep generative model composed of many RBM layers of latent variables and has potential for learning internal descriptions of data. The DBN can learn efficiently using feature activations with large unlabeled data inputs and very limited labeled data. On the other hand, EEG dataset is labeled, so we need to modify the structure of DBN for classification accuracy.

We add a *discriminative* objective function on top of the existing DBN. There are now several ways for discrimination. First, one can use some standard discriminative methods which use features learned by DBN as inputs, for example, logistic regression, SVM [26]. We admit that this may be the most efficient method for the problem. However, for the time being, we do not use it to make the model construction more convenient. Second, one can modify the *generative* DBN model into a *discriminative* DBN model [27]. We use this mothod in this work, as shown in Fig 3.3. To be more specific, we train RBM on each class (we have only two groups: epileptic spike and non spike), and then obtain the free-energy of a test data vector for each class. The free energy of a visible vector ($\mathrm{F}(\mathbf{v})$) is defined as the energy a configuration need to obtain in order to have same probability as all configuration that contain $\mathbf{v}$ [27].

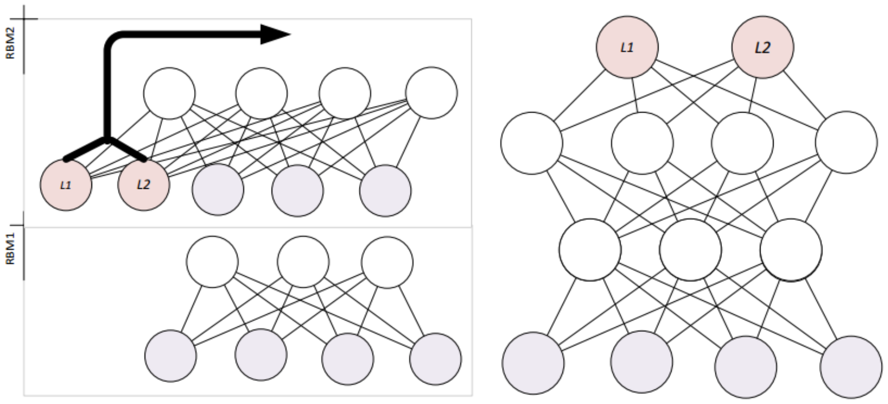


Figure 3.3: Discriminative Deep Belief Network

For each class-specific RBM, we have that

$$e^{-F(\mathbf{v})} = \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{3.1}$$

$$F(\mathbf{v}) = -\sum_i v_i a_i - \sum_j p_j x_j + \sum_i (p_i \log p_i + (1-p_i)\log(1-p_i)). \tag{3.2}$$

It is also calculated by

$$F(\mathbf{v}) = -\sum_i v_i a_i - \sum_j \log(1 + e^{x_j}) \tag{3.3}$$

where $x_i = b_i + \sum_i v_i W_{ij}$ is the total input to hidden unit j, $p_j = \sigma(x_j)$ is the probability that $h_j = 1$ give $\mathbf{v}$.

Recall that there are only 2 classes in EEG data, so it is easy to predict the probability of assigning a vector to a class from the free energies as

$$p(class = c|t) = \frac{e^{-F_c(t)}}{\sum_{d=1}^{2} e^{-F_d(t)}} \tag{3.4}$$

where Z_c is partition function as mentioned in the previous chapter, $F_c(t)$ is a free energy of the test vector t on class c.

How many layers and units is optimal?

A natural question would be: *what is (1) the optimal number of hidden layers and (2) the optimal number of units within a layer so as to achieve the best performance?*. In this part, we try to answer this question and give our choice.

It may be intuitive that if the DBN has many more hidden layers, the network is able to learn more complex features in data with high accuracy. However, this can be misconception. We will argue this through the following example. In [28], Michael Nielsen, who was classifying MNIST data using deep neural networks, investigated how increasing the number of layers can affect the performance. He first used one hidden layer for training (then the total system contains input layer - a hidden layer - output layer), and the classification accuracy was good. He then added another hidden layer (with same number of units to the first layer) and got a better result. As far, the more depth was good; hence, he added another layer with encouragement. Suddenly, the result fell down to be close to his original shallow network. One more time, he tried inserting more layers into the deep network, but it was not encouraging, either. He indicated that it is really difficult to get exactly the number of layers for the best results.

The degrading performance when the network is deeper can be explained by the *overfit* and/or *vashing gradient*. The former is that a model tailors to fit for some specific sample in the data but does not reflect the overall data [29], so the model would not fit for new data. The latter problem is that when signal moves backward through layers, non-linear functions at each stage make it very difficult for input layer to know exactly what is output target. These problems reduce the accuracy of the prediction and lengthen the learning time. Therefore, a large number of layers may be not good. On the other hand, it is trivial that the ability of DBN is limited if it has too few layers, preventing it from learning features of complicated data. In short, there is no absolute answer to how many hidden layers a deep neural networks should have. It does depend on the type of dataset. Thus, we have the answer for the first half of the question.

Now, the second half of the question concerns the number of hidden units. Again, it is difficult to have an absolute answer. It depends on the type and architecture of data. Too few and too many neurons can both degrade the classification result.

We configure the DBN, according to [27], as the following. The number of units in input and output layer corresponds to the true length of vector feature input and possible classifications on EEG data. The number of units in each hidden layers will be tested in simulation to find the best number of hidden units. Besides, we also let the number of hidden layers vary. Those settings of number of layers and number of units constitute several configurations of the DBN. We will test these configurations to examine the best deep architecture of DBN for our EEG dataset. The testing scenario will be shown in Chapter 4.

Learning rate

In any CD (gradient-based learning) algorithms, the learning rate ε is critical. A small learning rate will slow down the speed of convergence. On the other hand, a large learning rate usually increases the reconstruction error significantly and possibly leads to overfit. In [27], Hinton recommends a rule for setting learning rate by generating histograms of weight updates and weight values. The author suggests to adjust the learning rate to scale the weight updates to about 10^{-3} times, though provides no further justification for that choice. In our experiments, the learning rate is selected randomly and examined the best choice.

Training

As in a typical DBN, the training process of our DBN consists of three stages as follows.

- Pre-training each layer in a greedy way: We randomly initialize the parameters of the network and then apply Contrastive Divergence to update the hidden layer and visible layer. This so-called positive and negative phases make the probability of training data increase and decrease, respectively.
- Using unsupervised learning at each layer and training whole DBN: The weights are updated by taking the derivative of the probability of visible units with respect to weights;
- Fine-tuning the whole network: All the parameter of the whole network are fine-tuned using back-propagation and gradient descent on the cost function to obtain the optimal results [28].

Chapter 4

RESULTS

4.1 EEG Dataset

The EEG data used in this study were recorded at Signal and Systems Laboratory using the international standard 10-20 system with 19 channels and the sampling rate of 256 Hz. The measurements were carried out on patients aged from 6 to 18 years. In our experiment, the number of training data is 767 peaks including 285 epileptic spikes and 482 non-spikes obtained from 2 patients. The data is divided into two groups: training set and validation set containing 667 and 100 peaks, respectively. The testing data is from a channel of a patient with 78 epileptic spikes.

In order to obtain the EEG signal within the desired frequency band and restrain the noise due to ECG, EOG, EMG, which are acquisition of electrical activity of the heart, human eye and skeletal muscles respectively, we filter the EEG data with a digital Butterworth low-pass filter with cut-off frequency 70 Hz and order 5; a Notch filter with cut-off frequency 50 Hz, bandwidth 2 Hz, and order 4; and a high-pass filter with cut-off frequency 0.5 Hz and order 1.

4.2 Evaluation Metric

Accuracy, sensitivity and specificity are typical statistical measures of the performance of a classifier. We use them to evaluate the quality of the multi-stage system due to our improvement.

The accuracy of the classifier is computed by the following equation:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4.1}$$

The sensitivity measures the proportion of positives that are correctly identified and is computed as

$$SEN = \frac{TP}{TP + FN}. \tag{4.2}$$

The specificity measures the proportion of negatives that are correctly identified and is computed as

$$SPE = \frac{TN}{TN + FP}. \tag{4.3}$$

In the above formulas of accuracy, sensitivity and specificity, we let TP and FP denote the number of correctly identified and incorrectly identified samples, respectively; TN, FN denotes the correctly rejected and incorrectly rejected samples, respectively.

Moreover, we also use ROC curve to illustrate the performance of the system. The curve is drawn by plotting true positive rate (which is the Sensitivity SEN defined above) and false positive rate that can be calculated as $1 - SPE$. ROC analysis allows us to have a cost/benefit figure of decision-making.

4.3 Results

The multi-stage system is implemented in MATLAB 2015b on Ubuntu 15.10, running on a 4G-RAM and i5 intel processor using Deep Learning Toolbox given in [30]. In the experiments, training DBN is performed through 3 steps, as specified in Chapter 3: (1) pre-training of each layer, (2) training all layers and (3) fine-tuning of all with back-propagation. The goal of the training is to learn the weights and biases between each layer and reconstruction so that the input are as close to their output as possible.

Several different configurations of the DBN in terms of the number of hidden layers and hidden units are tested to choose the best result. First, three different hidden layer configurations in DBN are trained. After that, we change the number of neurons in each hidden layers and then decide the best structure of DBN for epileptic spike detection problem. The results are shown statistically in the Tab. 4.1 to 4.5. In particular, Tab. 4.1 gives the result for varying number of hidden layers and fixed number of hidden units, while the four next tables gives the result for fixed number of hidden layers and varying number of units in each or every hidden layer. The DBN configurations are shown as [input layer, hidden layers, output layer].

Table 4.1: Different number of layers

DBN	Events	Spikes	TP	FN	TN	FP	ACU
[224:30:2]	1797	76	52	24	1436	285	82.3%
[224:30:30:2]	1797	76	65	11	1453	268	84.47%
[224:30:30:30:2]	1797	76	70	6	1511	210	87.97%
[224:30:30:30:30:2]	1797	76	65	11	1489	232	86.47%

From Tab. 4.1, it can be seen clearly that the deep learning architecture with five layers (1 input, 1 output, and 3 hidden layers) allows us to have the best classification accuracy.

Table 4.2: Different number of hidden units in first hidden layer

DBN	Events	Spikes	TP	FN	TN	FP	ACU
[224:30:30:30:2]	1797	76	70	6	1511	210	87.97%
[224:100:30:30:2]	1797	76	68	8	1546	175	89.81%
[224:500:30:30:2]	1797	76	66	10	1449	272	84.31%
[224:1000:30:30:2]	1797	76	69	7	1459	262	85.03%

Table 4.3: Different number of hidden units in second hidden layer

DBN	Events	Spikes	TP	FN	TN	FP	ACU
[224:1000:30:30:2]	1797	76	69	7	1459	262	85.03%
[224:1000:300:30:2]	1797	76	71	5	1564	157	90.98%
[224:1000:500:30:2]	1797	76	70	6	1547	174	89.98%
[224:1000:1000:30:2]	1797	76	70	6	1462	259	85.25%

Table 4.4: Different number of hidden units in last hidden layer

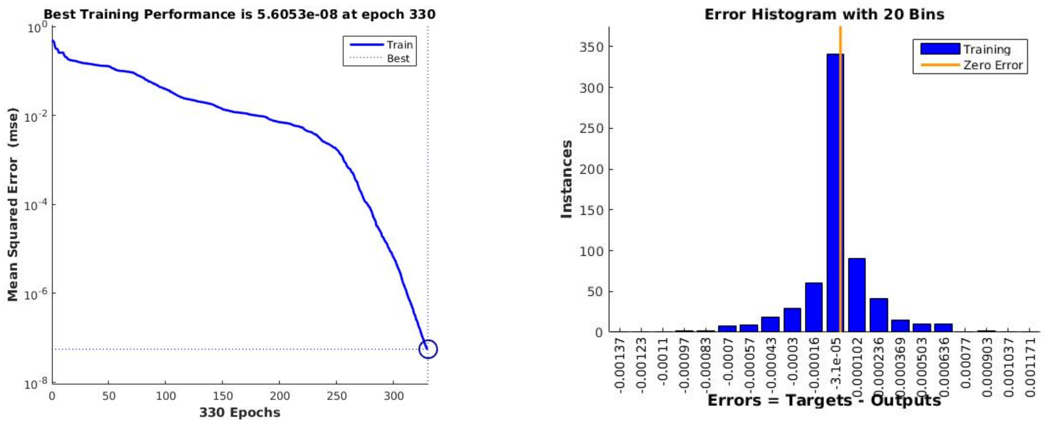
DBN	Events	Spikes	TP	FN	TN	FP	ACU
[224:1000:30:30:2]	1797	76	69	7	1459	262	85.03%
[224:1000:30:300:2]	1797	76	67	9	1492	229	86.75%
[224:1000:30:500:2]	1797	76	65	11	1579	142	91.48%
[224:1000:30:1000:2]	1797	76	69	7	1511	210	87.9%

Table 4.5: Different number of hidden units

DBN	Events	Spikes	TP	FN	TN	FP	ACU
[224:30:30:30:2]	1797	76	70	6	1511	210	87.97%
[224:100:50:30:2]	1797	76	70	6	1483	238	86.42%
[224:1000:300:30:2]	1797	76	71	5	1564	157	90.98%
[224:1000:500:100:2]	200	76	68	8	1499	222	87.2%

Table 4.6: Quantitative statistics using [224:1000:300:30:2] DBN

Patient	All	Preprocessing	DBN
1 channel	194832	1797	228



(a) (b)

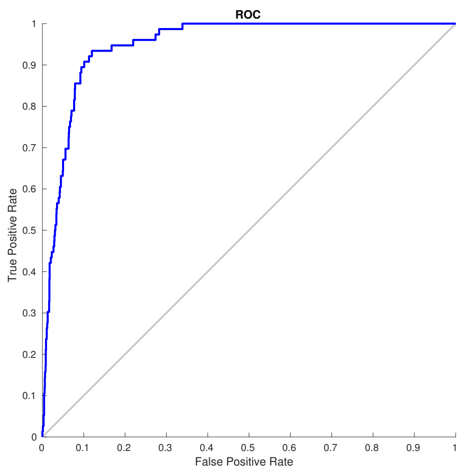
Figure 4.1: The error measures of DBN: (a) mean square error; (b) error histogram.

Next, the results for the cases of varying number of units confirm that the number of units should be under a threshold for each layer to obtain best results. If they overcome this value, the classification accuracy will drop . This negates the intuition that the more number of neurons in each layer, the more efficient performance. By comparing across tables, we observe that the configuration [224:1000:300:100:2] of DBN has the highest average performance in item of sensitivity, specificity, and accuracy 93.42%,90.87% and 90.98% respectively. Although this configuration requires a large number of neurons, it does not take a long time to train data if we adjust epochs to decrease the speed of convergence of CD algorithm.

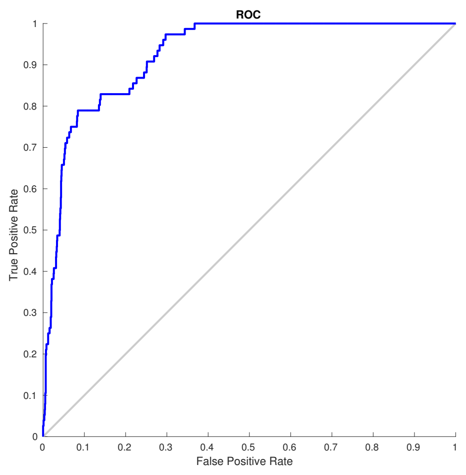
Training process using DBN indicates that learning high-level representations of EEG data can be achieved successfully for spike detection. In particular, the mean square error and error histogram, as shown in Fig. 4.1, are approximately 10^{-8} and 10^{-5}, respectively. Due to the non-stationary and complexity of epileptic spikes, although 667 peaks for training are really not *big* enough for any classifier to capture true spikes, the testing results included in five tables above are acceptable.

Table 4.7: A performance comparison between using DBN and ANN

	Events	Spikes	TP	FN	TN	FP	SEN	SPE	ACC
DBN	1797	76	71	5	1564	157	93.42%	90.87%	90.98%
ANN	1797	76	64	12	1401	320	84.57%	81.40%	81.5%



(a)



(b)

Figure 4.2: ROC curve of the system with: (a) Deep Belief Network; (b) Artificial Neutral Network.

In addition, we compare the performance of Deep Belief Networks with the shallow architecture - ANN used in [12]. The results are show statistically in Tab 4.7. It is clear that all the quality evaluation, namely sensitivity, specificity, and accuracy of DBN are better than that of ANN. The ROC curve on real testing data, as shown in Fig. 4.2, is also better for DBN than ANN. Moreover, using DBN consumes less training time than using ANN for two reasons. First, we are using DBN with only 5 layers and smaller number of neurons in DBN while [12] uses ANN with 41 layers. Second, the training time of DBN can be reduced by the decreasing the number of iterations to convergence in CD algorithm. The sensitivity, selectivity, and accuracy of the DBN classifier are all over 90% and around 10% better than the classifier ANN. This emphasizes the advantage and efficiency of DBN in epileptic spikes detection.

Chapter 5

CONCLUSIONS

In conclusion, we have applied the DBN model as a classifier to detect epileptic spikes in EEG signal. Before training EEG data using the DBN, we combine multi-step of preprocessing stage including small spike elimination, filtering, perceptrons and then propose a new scale wavelet range (32 steps from 3 to 9) in feature extraction stage. Training process show that the DBN can learn hidden features in EEG data which distinguish between epileptic spikes and non spikes group with high accuracy. The experiment results also indicate that learning high-level feature representations using the DBN model obtain better performance than those earlier methods like ANN.

In near feature, we would like to build a new model to get more suitable features for deep networks with better classification result. We will also continue to complete the DBN model and try to use the other deep learning models, adjust the parameters of these networks to determine which is the best model for detecting spikes in EEG signal. Moreover, to get higher quality, we will consider improving pre-processing with more appropriate design of filters, perceptrons to get clearer data before training deep learning models.

References

[1] "Epilepsy Society," https://www.epilepsysociety.org.uk.

[2] "Epilepsy Foundation," http://www.epilepsy.com/learn.

[3] "WHO," http://www.who.int/mediacentre/factsheets/fs999/en.

[4] Ö. Özdamar and T. Kalayci, "Detection of spikes with artificial neural networks using raw EEG," *Computers and Biomedical Research*, vol. 31, no. 2, pp. 122–142, 1998.

[5] T.-W. Shen, X. Kuo, and Y.-L. Hsin, "Ant K-Means Clustering Method on Epileptic Spike Detection," in *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, vol. 6. IEEE, 2009, pp. 334–338.

[6] N. Acır and C. Güzeliş, "Automatic spike detection in EEG by a two-stage procedure based on support vector machines," *Computers in Biology and Medicine*, vol. 34, no. 7, pp. 561–575, 2004.

[7] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[8] P. Mirowski, D. Madhavan, Y. LeCun, and R. Kuzniecky, "Classification of patterns of EEG synchronization for seizure prediction," *Clinical neurophysiology*, vol. 120, no. 11, pp. 1927–1940, 2009.

[9] W.-L. Zheng and B.-L. Lu, “Investigating critical frequency bands and channels for EEG-based emotion recognition with deep neural networks,” *Autonomous Mental Development, IEEE Transactions on*, vol. 7, no. 3, pp. 162–175, 2015.

[10] D. Wulsin, J. Gupta, R. Mani, J. Blanco, and B. Litt, “Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement,” *Journal of neural engineering*, vol. 8, no. 3, p. 036015, 2011.

[11] S. M. Plis, D. R. Hjelm, R. Salakhutdinov, E. A. Allen, H. J. Bockholt, J. D. Long, H. J. Johnson, J. S. Paulsen, J. A. Turner, and V. D. Calhoun, “Deep learning for neuroimaging: a validation study,” *Frontiers in Neuroscience*, vol. 8, 2014.

[12] A.-D. T. Nguyen, N. Linh-Trung, T. Tran-Duc, H.-A. T. Nguyen, and B. Boashash, “A multistage system for automatic detection of epileptic spikes,” 2015.

[13] E. D. Übeyli and İ. Güler, “Features extracted by eigenvector methods for detecting variability of EEG signals,” *Pattern Recognition Letters*, vol. 28, no. 5, pp. 592–603, 2007.

[14] Y.-C. Liu, C.-C. K. Lin, J.-J. Tsai, and Y.-N. Sun, “Model-based spike detection of epileptic EEG data,” *Sensors*, vol. 13, no. 9, pp. 12 536–12 547, 2013.

[15] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” *Neural computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[16] H. S. Liu, T. Zhang, and F. S. Yang, “A multistage, multimethod approach for automatic detection and classification of epileptiform EEG,” *Biomedical Engineering, IEEE Transactions on*, vol. 49, no. 12, pp. 1557–1566, 2002.

[17] N. Yadav, A. Yadav, and M. Kumar, *An Introduction to Neural Network Methods for Differential Equations.* Springer, 2015.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in neural information processing systems*, 2006, pp. 1345–1352.

[20] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 129–136.

[21] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[22] G. Hinton and M. A. Carreira-Perpinan, "On Contrastive Divergence Learning." in *AISTATS*, vol. 10. Citeseer, 2005, pp. 33–40.

[23] R. Salakhutdinov, "Learning Deep Generative Models," *Annual Review of Statistics and Its Application*, vol. 2, pp. 361–385, 2015.

[24] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[25] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[26] Y. Tang, “Deep learning using linear support vector machines,” *arXiv preprint arXiv:1306.0239*, 2013.

[27] G. Hinton, “A practical guide to training restricted Boltzmann machines,” *Momentum*, vol. 9, no. 1, p. 926, 2010.

[28] M. Nielsen, “Neural Networks and Deep Learning,” http://neuralnetworksanddeeplearning.com/, Jan 2016.

[29] M. A. Babyak, “What you see may not be what you get: a brief, nontechnical introduction to overfitting in regression-type models,” *Psychosomatic medicine*, vol. 66, no. 3, pp. 411–421, 2004.

[30] M. A. Keyvanrad and M. M. Homayounpour, “A brief survey on deep belief networks and introducing a new object oriented MATLAB toolbox (DeeBNet V2. 1),” *arXiv preprint arXiv:1408.3264*, 2014.