ABSTRACT

Title of thesis: TIME SERIES ANALYSIS USING DEEP FEED FORWARD NEURAL NETWORKS

Jeffrey T. Turner, Master of Science, 2014

Thesis directed by: Professor Tim Oates
Department of Computer Science
and Electrical Engineering

Deep neural networks can be used for abstraction and as a preprocessing step for other machine learning classifiers. Our goal was to develop methods for a more accurate automated seizure detection. Deep architectures have been used for classification of events, and shown in this research to be an effective way of classifying multichannel high resolution medical data. The medical data used in this thesis was gathered from an electroencephalograph (EEG) used in a hospital setting on seizure patients.

To demonstrate the ability of deep architectures to learn and abstract from input data, the signals from the EEG that contained both seizure and non seizure data were given both as featurized data and raw data to the deep architecture. In addition to the multiple types of data preparation, a patients EEG data was tested not only against their own EEG signal training data but other patients as well. This study supports the effectiveness of deep feed forward neural networks for usage in the seizure classification scenario, as well as highlights some of the difficulties associated with training deep neural networks, as shown through experimental results.

TIME SERIES ANALYSIS USING DEEP FEED FORWARD NEURAL NETWORKS

by

Jeffrey T. Turner

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County in partial fulfilment
of the requirements for the degree of Master of Science 2014

Dedication

For Andrea:

The sooner I'm done with this thing the sooner we can get a kitten. I love you.

Acknowledgments

I would like to thank all of the people who have helped me, while specifically ensuring that I do so in no particular order, as to not offend someone by accident. To help facilitate this, I generated the list components, and modified their order with a cryptographically secure random number generator multiple times, while simultaneously poking the cat. When the cat moved, I used that configuration of acknowledgements. I would like to thank:

- Firstly Andrea. She didn't get randomized, because she's the reason I want to get this thesis done ASAP or sooner. You have heard me complain and stress so much about deep learning you probably never want to hear it again. Thank you for everything you mean to me, and all the help you gave me in this thesis! I love you.

- Secondly, I would like to thank my family. You've supported me through the years, even when there wasn't much of me left to support. If I could beat any of you at fantasy hockey, it would be the perfect family.

- Next, I would like to thank the Energy Efficient High Performance Computing Lab members of Adam and Tinoosh that we collaborated with getting published at AAAI Spring Symposia and FLAIRS too! We all put in a tremendous amount of work over the past year, and I got my first published paper out of it!

- Dr. Tim Oates. You are so not the professor from phdcomics, and for that I am very grateful. You showed a lot of faith in me to get work done when you hired me, and I hope I haven't let you down. I'll beat you next race.

- The CoRaL lab; a bunch of really smart guys (and a gal), and then me. I enjoy all of our conversations, whether it be about machine learning, baseball, or whatever else comes up that day.

- A series of excellent teachers I've had over many years. From college this list INCLUDES (but is not limited too): Professor David Daniel (who recently recieved the prestigous award for a second time of *hottest professor in the nation* according to ratemyprofessor), Professor Laura Taalman (who taught me math is relevant, and how to LaTeX), Professor Sue Evans (I STILL used Python for my masters thesis!), Professor Ryan Bergeron (The caps will win the cup, keep watching), Professor Tim Oates (You already got a whole paragraph), Professor Anissa Sorokin (who I asked around a quadrillion grammar questions in writing this), and Professor Anupam Joshi (Someone once told me my aim as an undergraduate should be to take as many classes with Dr. Joshi as possible, and they were right).

- Finally I would like to thank the crew team, with a special emphasis to the boat I 2k'd in (Kpop, Catman, Zach and Andy). Really though I'd like to say thanks to the whole crew team except Griffin for being good friends and teammates. OK, maybe Griffin too.

So long UMBC, and thanks for all the fish.

Table of Contents

List of Figures

List of Abbreviations

α	alpha (Learning Rate)
β	beta (Denoising Factor)
θ	theta (Parameters)
Δ	Delta (Parameter Shift)
τ	tau (Seizure Threshold)
F_1	F Measure
ℓ	Loss Function
$\bar{x}$	Mean
σ	Standard Deviation

BBN	Bayesian Belief Network
CD-k	Contrastive Divergence of k steps
dA	Denoising Autoencoder
CHB	Childrens Hospital of Boston
CSV	Comma Separated Values
DBN	Deep Belief Network
EDF	European Data Format
EEG	Electroencephalograph
EBM	Energy Based Model
GPED	Generalized periodic epileptiform discharges
GPU	Graphics Processing Unit
LOO	Leave One Out
LR	Logistic Regression
MNIST	National Institute for Standards and Technology Digit Classification
PLED	Periodic lateralized epileptiform discharges
RBM	Restricted Boltzmann Machine
SdA	Stacked Denoising Autoencoders
SVM	Support Vector Machine

Chapter 1: Introduction

This chapter will provide a high level overview of the scope of the research, and outline the thesis.

1.1 Time Series and Deep Learning

The task of developing a representation of data that is effective for use in machine learning applications is difficult, and affects the final classification results. It should be carefully planned to maximize accuracy. Time series data introduces the additional problem of being able to represent the data in a form that can can also convey changes over time. In the medical setting, which is the focus of this study, the *Electroencephalograph* (EEG), which measures strength of electrical signals by scalp electrodes. The EEG is multichannel, and produces time series of the readings over the various channels through time. Extra considerations for the multiple channels are required in both training and testing; although the addition of extra channels makes the computation and design of the algorithms more complex, there is an added power of expressiveness gained by sampling multiple views of the data. Feed forward deep neural networks can be trained in such a way that they are effective at representing many kinds of data, whether they are raw data, or featurized.

In this thesis, I use the *Stacked Denoising Autoencoder* (SdA) and *Deep Belief Networks* (DBN). Problems containing Time series data behave similar to many other machine learning and data mining problems such that there is no master algorithm that is the best for all problems, but many different representations of data exist and must be considered [1].

1.2 Seizures

Seizures are neurological conditions where irregular electrical signals are generated on the surface of the brain and cause symptoms for the seizure victim. The symptoms associated with these disturbances can be as mild as a hand tremble, or as severe as loss of consciousness and full body spasms [2]. Seizure and other brain activity produces electrical signals in the nerve cells of the cerebral cortex, and are recorded via an EEG. With the current state of understanding in seizures and signal processing from EEGs, prediction of seizures is still an unsolved and difficult problem[3]. Machine based detection of seizures is a hard task, and therefore is for the most part still done by manual inspection of EEG readings in clinical settings [4]. As will be explained in much greater detail in the following chapter, the multiple channels of the EEG measure electrical impulses at different parts of the brain. Techniques for representing multi-channel time series data will be explained in great detail chapter in 2. This thesis explores the task of seizure detection in a clinical setting, where a high resolution multi channelled EEG is being used on patients.

1.3 Overview of Approach and Results

EEG data from patients undergoing observation for seizures was prepared, represented, and grouped in a variety of different manners for input to the deep learning architectures. For the experiments, either an SdA or a DBN were the target for the data, with a *Logistic Regression* (LR) classifier taking the final output of the deep architecture to do classification. In the first dataset, features were extracted from the raw data using various mathematical properties of continuous waves; these features have been previously shown to be an effective means of EEG analysis [4], and will be expanded upon in greater detail in Chapter 2. The results were measured on the basis of F_1 measure (the harmonic mean of precision and recall) instead of classification accuracy because of the small fraction of data that was actually a seizure. Using this simple feature set and a DBN, a high F_1 measure was obtained, and the same feature set was also fed as input to a LR classifier with no DBN preprocessing as a baseline. As is discussed in chapter 4, performance was generally better when the data was run through a DBN before being fed into LR than when data was passed straight to the LR, but not always. Because deep architectures are proficient in abstraction of data [5], testing was also done where training and validation was performed on one group of patients but final classification accuracy was performed on a patient not seen in any of the training or validation examples. Once again the DBN performed well, and generally better than only using LR.

For a problem more suited to deep architectures, the second experiment did not involve extracting features from the data; raw data was used as input to the architecture [6]. An SdA was used as opposed to a DBN for this experiment, because the F_1 measure was consistently higher using an SdA on raw data. In addition to using an SdA, all of the channels were classified separately before an ensemble method was used to give a final class label to the second as a seizure occurring or not. The SdA performed well at this task, although the F_1 measure was lower for raw data versus featurized data fpr all patients tested with raw data. Finally, a situation where the training and validation patients were different from the testing patient was used as well as input to the DBN with no feature extraction or preprocessing. This difficult classification test showed lower F_1 measures on many of the classification tasks that other algorithms had excelled at, but did perform slightly better on some of the "difficult" patients where traditional feature extracted classification performed poorly.

1.4 Contributions

This research introduces two concepts to the realm of deep learning. The first is using deep neural networks such as DBNs and SdAs to classify EEG data as seizure or non seizure data. Although the work of EEG and seizure analysis using DBNs has been done before [4], the use of SdAs is new, and this study is classifying seconds of EEG data as seizure and nonseizure, while the other study attempts to identify various signals that are visible on an EEG such as an eye blink. Furthermore, this thesis uses a method of non-specific patient testing by means of training and validating on a corpus of patients, and testing on another. In contrast to previous work [7], this study elaborates using the non-specific patient testing as a way of creating channel ensembles.

1.5 Outline of Thesis

The remainder of the thesis is outlined as follows: Chapter 2 contains the needed background information to understand the remainder of the thesis. Firstly, how time series and multi channel time series are used in this study are discussed, and the features extracted as input to the DBN are enumerated. Next, the DBN and SdA algorithms are explained in more detail, which will require an understanding of *Restricted Boltzmann Machines* (RBMs), and *Denoising Autoencoders* (dAs). The EEG that was used for the collection of the data is described, and finally in Chapter 2 related works in deep networks and seizure detection are referenced.
Chapter 3 consists of details describing the dataset used in the experiments. Firstly, a description of the source of the data is given, and then a detailed description of how the data was prepared and grouped for input to the algorithms is given as well.
Chapter 4 is the results of all of the experiments run. This includes:

- Single Patient Featurized DBN. In this experiment, a single patients featurized EEG data is given as input to a DBN for training, and a different set of EEG data from the same patient is given to the DBN for testing.

- Single Patient Featurized LR. In this experiment, a single patients featurized EEG data is given as input to a LR for training, and a different set of EEG data from the same patient is given to the LR for testing.

- Leave One Out Training Patient Featurized DBN. In this experiment, featurized EEG data from all of the patients in the testing corpus minus one are given to a DBN for training. For testing, featurized data from the one patient not in the training group is used.

- Leave One Out Training Patient Featurized LR. In this experiment, featurized EEG data from all of the patients in the testing corpus minus one are given to a LR for training. For testing, featurized data from the one patient not in the training group is used.

- Single Patient Raw SdA Ensemble. For this experiment, raw EEG data from one patient is given to a SdA for training. A different set of raw EEG data from the same patient is used for testing.

- Leave One Out Training Patient Raw SdA Ensemble. In this experiment, raw EEG data from all of the patients in the testing corpus minus one are given to a SdA for training. For testing, raw data from the one patient not in the training group is used.

In addition to showing experiment results, Chapter 4 draws attention to some of the more subtle points of the results that are easy to miss at first glance, as well as highlight some of the strengths and weaknesses of the study. Chapter 5 consists of a conclusion and closing thoughts.

Chapter 2: Background

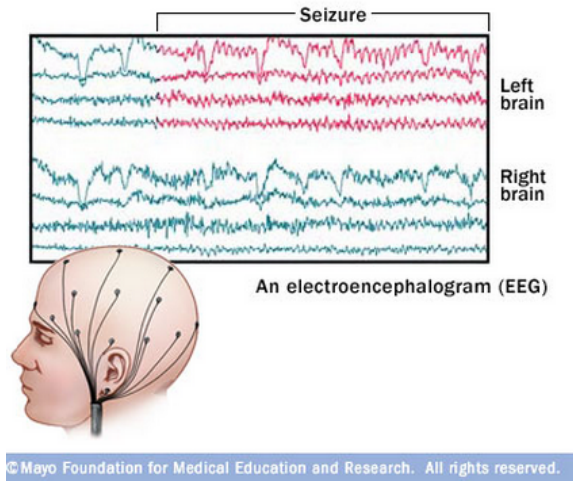
In this chapter, I will give the needed background information to understand the remainder of this research.

2.1 Electroencephalography Data

The *Electroencephalograph* (EEG) is a device used to monitor electrical activity on the surface of the brain; usually anomalous conditions such as seizure or sleep disorders, but it also has limited applications in brain computer interfacing [8]. While the data is often shown as continuous wave forms (which is how humans are able to detect seizures and identify other brain activities from the EEG readings), the data that is received by the machine itself is many discrete electrical readings measured in millivolts (mV).

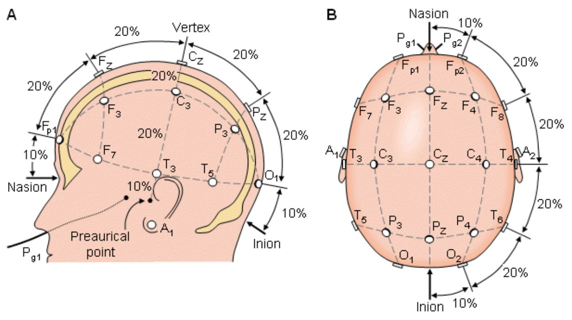
Depending on the design of the actual system itself, the number of readings per second (Hz) varies. The high resolution clinical EEG that is used in the thesis measures 23 channels at 256 Hz making time series methods appropriate for seizure classification tasks. The EEG channels are placed using the international 10-20 system for electrode placement shown in Figure 2.1. The CHB-MIT dataset [7] had 26 patients, however only 9 of them were used in the thesis. These 9 had the same

Figure 2.1: EEG Data Indicating a Left Brain Seizure


Seizure
Left brain
Right brain
An electroencephalogram (EEG)
©Mayo Foundation for Medical Education and Research. All rights reserved.

number of scalp electrodes and the same position; from this point on when I refer to all of the patients, I will be referring to the 9 patient set, not the 27 patient set.

Figure 2.2: International 10-20 System for scalp electrodes



2.1.1 Multi Channel Time Series Data

Time series problems are prevalent in diverse domains such as finance, medicine, industrial process control, and meteorology [1]. Let T be a time series representing an EEG that samples at a rate of H Hz over C channels for S seconds. The multi-channel time series can be denoted as follows (where $X_{c,s,h}$ is the reading of the c^{th} channel on the h^{th} sample of the s^{th} second, measured in mV):

$$
\begin{aligned}
T = \Big((\quad & (x_{0,0,0}, && x_{0,0,1}, ..., x_{0,0,H-1}), \\
& (x_{1,0,0}, && x_{1,0,1}, ..., x_{1,0,H-1}), \\
& \quad ... \\
& (x_{C-1,0,0}, && x_{C-1,0,1}, ..., x_{C-1,0,H-1})), \\
& \quad ... \\
(\quad & (x_{0,S-1,0}, && x_{0,S-1,1}, ..., x_{0,S-1,H-1}), \\
& (x_{1,S-1,0}, && x_{1,S-1,1}, ..., x_{1,S-1,H-1}), \\
& \quad ... \\
& (x_{C-1,S-1,0}, && x_{C-1,S-1,1}, ..., x_{C-1,S-1,H-1}))\Big).
\end{aligned}
$$

This results in a massive amount of high resolution clinical quality EEG data that is so large that it is unlikely to perform well on classification tasks using a traditional machine learning algorithm such as Decision Trees, and would quickly become far too large of a model for usage with K-Nearest Neighbor (KNN) algorithms, especially without featurization. As shown later, deep neural networks have the ability to handle large featurized data or raw data which is how the data is initially presented in the problem.

2.2 Feature Set

The following feature set was used with success in other work with EEG data [4]. I will define some naming conventions to be used throughout the section. A *peak* is defined as a reading that marks the change from a positive to negative derivative, and a *valley* is defined as a reading that marks a change from a negative derivative to a positive derivative. k_i will mark the index of the k^{th} peak of the time series, with a value of $x_{k(i)}$. Similarly, v_i will mark the i^{th} valley index, and $x_{v(i)}$ will denote the value of it. Window size is given by W. Given a single channel time series T, with reading $x_i \forall i \in T$ where i is the index.:

- *Area*: Area under the wave for the given time series. Computed as:

$$A = \frac{1}{W} \sum_{i=0}^{W-1} |x_i|.$$

- *Normalized Decay*: Chance corrected fraction of data that has a positive or negative derivative. $I(x)$ is a boolean indicator function, whose value is 1 when true, 0 when false.

$$D = |\frac{1}{W-1} \sum_{i=0}^{W-2} I(x_{i+1} - x_i < 0) - .5|.$$

- *Line Length*: Summation of distance between all consecutive readings.

$$\ell = \sum_{i=1}^{W-1} |x_i - x_{i-1}|.$$

- *Mean Energy*: Mean energy of time interval.

$$E = \frac{1}{W} \sum_{i=0}^{W-1} x_i^2.$$

- *Average Peak Amplitude*: Log base 10 of mean squared amplitude of the K peaks.

$$P_A = \log_{10}\Big(\frac{1}{K}\sum_{i=0}^{K-1} x_{k(i)}^2\Big).$$

- *Average Valley Amplitude*: Log base 10 of mean squared amplitude of the V valleys.

$$V_A = \log_{10}\Big(\frac{1}{V}\sum_{i=0}^{V-1} x_{v(i)}^2\Big).$$

- *Normalized Peak Number*: Given K peaks, normalized peak number is the number of peaks normalized by the average difference between data readings.

$$N_P = K\Big(\frac{1}{W-1}\sum_{i=0}^{W-2} |x_{i+1} - x_i|\Big)^{-1}.$$

- *Peak Variation*: Variation between peaks and valleys across time (measured in Hz), and electrical signal (measured in mV). In the case where the number of peaks is not equal to the number of valleys (if a time interval begins or ends during an increase or decrease in data, it is not recorded as a peak or valley), then the feature with the least features is used in comparisons between peaks and valleys. The mean ($\mu(PV)$) and standard deviation ($\sigma(PV)$) of the indicies are given by:

$$\begin{aligned} \mu(PV) &= \tfrac{1}{K}\sum_{i=0}^{K-1} K_i - V_i. \\ \sigma(PV) &= \sqrt{\tfrac{1}{K-1}\sum_{i=0}^{K-1} (K_i - V_i - \mu(PV))^2}. \end{aligned}$$

The difference in readings is given by

$$\begin{aligned}\mu(x_{PV}) &= \tfrac{1}{K}\sum_{i=0}^{K-1} x_{k(i)} - x_{v(i)}.\\ \sigma(x_{PV}) &= \sqrt{\tfrac{1}{K-1}\sum_{i=0}^{K-1}(x_{k(i)} - x_{v(i)} - \mu(x_{PV}))^2}.\end{aligned}$$

The peak variation is calculated as

$$P_V = \frac{1}{\sigma(PV)\sigma(x_{PV})}.$$

- *Root Mean Square*: The square root of the mean of the data points squared.

$$R_{MS} = \sqrt{\frac{1}{W}\sum_{i=0}^{W-1} x_i^2}.$$

2.3 Logistic Regression

Logistic regression is a widely used machine learning algorithm, and is used in every one of our experiments, often times sitting on the output layer of the deep neural networks, but for control purposes it is also used as a benchmark for the feature set experimentation. The logistic regression algorithm with weight matrix W and bias vector b, predicts class label Y of training instance x with probability:

$$P(Y = i|x, W, b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}. \tag{2.1}$$

The classification function above defines a linear hyperplane separator as opposed to a logistic hyperplane as the name implies. For Logistic Regression, the loss function used where θ is the parametrization of the model applied to data set D is given by:

$$\ell(\theta, D) = -\sum_{i=0}^{|D|-1} log(P(Y = y^i|x^i, W, b)). \tag{2.2}$$

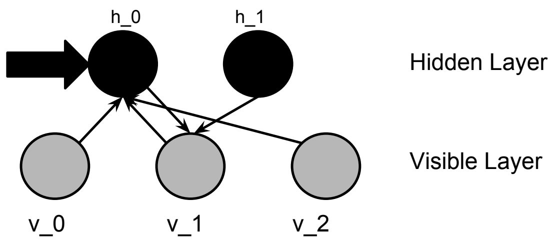
The logistic regression implementation was used from the Theano Project [9], with slight modifications for ability to save and load models.

2.4 Deep Feed Forward Neural Networks

In recent years, deep learning has shown great promise in tasks such as robotic vision, natural language processing, and data mining [10][11]. In 2006, deep neural networks became popular after the discovery that deep networks were useful when a saturating nonlinearity was used [12], as opposed to the case of the single hidden layer perceptron. With the use of *graphics processing units* (GPUs) it is possible to train deep artificial neural networks in a layer wise fashion to tackle problems that previously required discretization. The common element of all of the deep architectures is the usage of probability distributions to stochastically modify small parts of the input data during training. The reason this is done is to prevent the model from learning a 1-1 mapping of the data, as this overfitting of training data would inevitably hinder test performance [13]. Backpropogation is another challenge faced by deep feed forward networks, as evidence suggests that in the sigmoid output layer of deep architectures, convergence is not guaranteed to be good, fast, or exist at all [14].

Not to be overlooked in the discussion of deep feed forward neural networks is the abstract concept of a *node*. An incomplete view of a layer of a deep neural network architecture is shown in Figure 2.4. Nodes of the neural network are a summation of the multiplications of the input vector by the weight matrix W, added to the visible bias vector (b_v), or the hidden bias vector (b_h). b_v is used when calculating the visible nodes, while b_h is used when calculating the hidden nodes.

Figure 2.3: Deep Neural Network Architecture



The figure is incomplete, and only shows the calculations of the 0^{th} node of the hidden layer (h_0), and the 1^{st} node of the visible layer (v_1), as to not obstruct view of the arrows. Furthermore the figure is incomplete because the bias vector additions are not shown. Let us consider the node of the hidden layer h_0, which is pointed to by the large arrow. We abstract this node as a physical object, but if we look at the parameters of this neural network, we can see what the node actually consists of. For parameters (θ) consisting of W, b_v, and b_h, the parameters are as follows:

$$W = \begin{pmatrix} w_{0,0} & w_{0,1} \\ w_{1,0} & w_{1,1} \\ w_{2,0} & w_{2,1} \end{pmatrix} \quad , \quad b_h = (b_{h0}, b_{h1}) \quad , \quad h_v = (b_{v0}, b_{v1}, b_{v2})\,.$$

As is represented in the figure, the hidden node h_0 has incoming edges from all 3 of the visible layer nodes. For any arbitrary hidden node h_k, we can compute its value as follows:

$$h_k = (\sum_{i=0}^{|V|-1} W_{i,k} v_i) + b_{hk}, \tag{2.3}$$

and likewise any arbitrary visible node v_j can have its value calculated by:

$$v_j = (\sum_{i=0}^{|H|-1} W_{j,i}h_i) + b_{vj}. \tag{2.4}$$

In the remainder of this section we introduce terminology, review the deep learning methods used in this work, and discuss related work in the domain of seizure detection using machine learning.

2.4.1 Denoising Autoencoder

The autoencoder is a combination of a weight matrix W, a hidden bias vector b_h along with a visible vias vector b_v, and a saturating nonlinearity function (such as sigmoid or hyperbolic tangent) s, that can be applied to an input space x to recreate a coded version y as follows:

$$y = s(Wx + b_h). \tag{2.5}$$

The encoded version of y will be an alternate representation of the input vector x, possibly with a different dimension [10]. The sigmoid function is used in my implementation of dAs, as it has been shown to perform well for autoencoders [15]. From the representation y (which may or may not be a different dimensionality from x), the process can be reversed to reproduce the original input as:

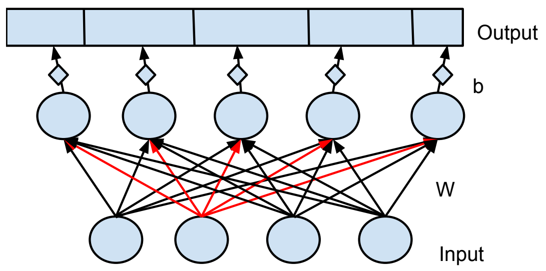
$$z = s(W^T y + b_v). \tag{2.6}$$

and assuming the weight matricies and bias vectors have been trained properly, $x = z$, because no noise or randomness was introduced. To make the autoencoder learn a deeper representation of the data, we introduce the idea of a *denoising autoencoder* (dA). The dA learns a representation of the input data by observing a slightly modified version of the input data to aid in the learning [16]. The probabilistically modified input data point z_i given input x, with corruption level of the input β is given by:

$$z_i = \begin{cases} 0 : & p = \beta \\ x_i : & p = 1 - \beta, \end{cases}$$

which is used to construct $Z := z_i \forall i \in |X|$. Through extensive trial and error, a β level that performed well for a large variety of input sizes and machine learning problems in robotic vision was found at $\beta = .1$, as recommended in the Theano manual [9].

Figure 2.4: Denoising Autoencoder



An example of a dA is shown in Figure 2.4.1, of input size 4 and output size 5. In the figure only the forward operation is shown; the 20 arrows between the circular nodes indicate the multiplication of the weight matrix W of dimension 4x5, and the 5 diamond arrows indicate the addition of the bias vector b. The red arrows between the circular nodes show the intentional modification of the data where the value given is zero. Not shown is the non linear sigmoid function that is applied to the output.

The loss function used for dAs is *cross entropy* (ℓ) to give a measure of distance between the true input data, and the reconstruction of the input data. Let x be the input to the dA, and x' be the denoised version of the input, with a noise probability of β.

$$y = s(Wx' + b_h) \tag{2.7}$$

$$z = s(W^T y + b_v) \tag{2.8}$$

Then we compute cross entropy as:

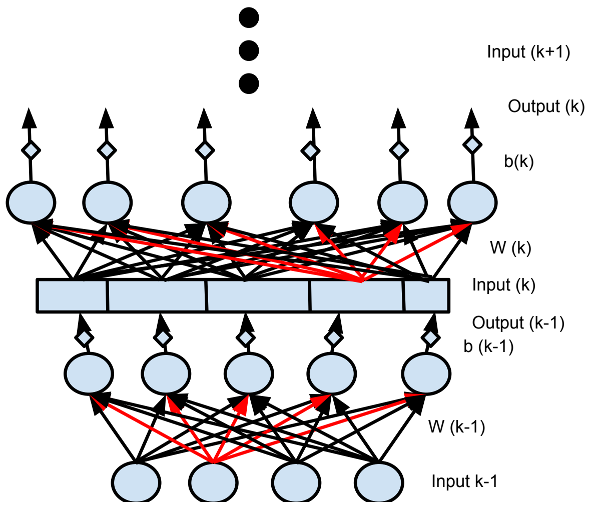
$$\ell(x, z) = -\sum_{i=0}^{|x|}(x_i log(z_i)) + (1 - x_i)log(1 - z_i). \tag{2.9}$$

Given a pre-training learning rate α, we can update the model with cost C as $C = \frac{\alpha\ell}{|x|}$. It is worth noting that in the cross entropy calculation we are comparing our reconstructed input z against the non corrupted input x so there will be some loss even in well trained functions for $\beta > 0$.

2.4.2 Stacked Denoising Autoencoder

Once the dA is in place as a working unit, we can begin to stack together chains of dAs to further the robustness of the algorithm. In a *Stacked Denoising Autoencoder* (SdA) consisting of N dAs, the output of dA $k - 1$ can be used as input to dA k, and the output of dA k can then be used as input to the following dA $k + 1$.

Figure 2.5: Stacked Denoising Autoencoder



The structure of the SdA is shown in Figure 2.4.2. The purpose of the dA is to provide abstraction from raw data, so by adding more dAs we are providing a way to abstract abstractions. The dropping out or zeroing out of both parts of the raw input data and the higher level abstactions as well has been shown to excel in learning tasks such as handwritten digit recognition [13].

The concept of SDAs is powerful and provides two important ideas implemented in this and other studies. First, the SDAs can be trained in a greedy manner using only one dA at a time [17]. Once we have trained the weight matrix W and the bias vector b for the layer, there is no need for W' or b' that were used to reconstruct the input, and we can disregard them. Secondly, the training of the dAs in the SDA does not require a class label for the loss function, meaning that the training can be done completely unsupervised. This fact helps to shed light on the purpose of SDAs and deep learning in general; the purpose of deep architectures is to learn the structure and abstractions of the underlying data useful in classification. The final layer of the SDA must be a classifier, since the SDA is only a preprocessor for the classifier. The classifier in this experiment is a logistic regression machine, with an input size equal to the output size of the highest dA, with outputs of the logistic regression classifier being “seizure” or “nonseizure”.

2.4.3 Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBMs) are a type of *Energy Based Model* (EBM); a preprocessing device that can represent dependencies between variables and infer values of hidden variables [18]. For all energy based models, the level of energy is related inversely to the probability of that configuration. This means EBMs find low energy configurations more plausible. The generic EBM probability function given input v is:

$$p(v) = \frac{e^{-E(v)}}{Z}, \tag{2.10}$$

where Z is the partition function, given by

$$Z = \sum_v e^{-E(v)}. \tag{2.11}$$

To increase performance of RBMs as well as simplify the math of the update rule, we can restrict ourselves to binary values for hidden and visible nodes. We can stipulate that all visible and hidden nodes at all time must maintain the state of 0 or 1. Given saturating non linearity s, the neuron activation functions where h is a hidden node layer, and b_v and b_h are the visible and hidden biases can be written as follows:

$$P(h_i = 1|v) = s(b_h + W_i v). \tag{2.12}$$

$$P(v_i = 1|h) = s(b_v + W_j^T h). \tag{2.13}$$

Because of the strict bipartite structure of the RBM, the hidden nodes are conditionally independent of each other given the other hidden nodes, and as the visible nodes are completely conditionally independent of each other as well given the other visible nodes, we have $P(v|h) = \prod_{i=0}^{|v|} P(v_i|h)$ and $P(h|v) = \prod_{i=0}^{|h|} P(h_i|v)$.

For RBMs, instead of expressing the energy as only a function of the input x, we make energy a function of v and h to represent the visible and hidden layers of the RBM. Since the input and hidden layers are vectors, the energy of an RBM given inputs v and hidden layer h is expressed as a matrix. In the following energy equation, b_v and b_h represent the visible and hidden biases, while W represents the weight matrix.

$$E(v, h) = -b_v^T v - b_h^T h - h^T W v. \tag{2.14}$$

Now, we can take the marginalization of energy to derive the *free energy* of the system in the log domain. This marginalization will be needed to calculate the gradients for update rules. Since it is a sum, the hidden vector is not needed as input, so we calculate $E_F(v)$ instead of $E_F(v, h)$.

$$E_F(v) = -b_v^T v - \sum_{i=0}^{|v|-1} \log(1 + e^{b_{hi} + W_i v}). \tag{2.15}$$

Although the topic of energy based probability models could fill a book on its own, their usage in this implementation of an RBM is much simpler. We will use one step of the contrastive divergence algorithm (CD-1), as this has been shown to be very effective in deep learning [10]. To justify the usage of contrastive divergence, let us consider the following derivation of an update rule for parameters θ with respect to the negative logarithmic probability of a given input x. Energy of the system is expressed as $E(x, y)$, with partition function given as Z.

$$\begin{aligned}
\frac{\partial}{\partial\theta}(-log(P(v)) = & \quad \frac{\partial}{\partial\theta}\left(- log \textstyle\sum_h P(v,h)\right) \\
= & \quad \frac{\partial}{\partial\theta}\left(- log \textstyle\sum_h \frac{e^{-E(v,h)}}{Z}\right) \\
= & \quad \frac{Z}{\sum_h e^{-E(v,h)}}\left(\textstyle\sum_h \frac{1}{Z}\frac{\partial e^{-E(v,h)}}{\partial\theta} - \sum_h \frac{e^{-E(v,h)}}{Z^2}\frac{\partial Z}{\partial\theta}\right) \\
= & \quad \textstyle\sum_h \left(\frac{e^{-E(v,h)}}{\sum_{\tilde{h}} e^{-E(v,\tilde{h})}} \frac{\partial E(v,h)}{\partial\theta}\right) + \frac{1}{Z}\frac{\partial Z}{\partial\theta} \\
= & \quad \textstyle\sum_h P(v|h)\frac{\partial E(v,h)}{\partial\theta} - \frac{1}{Z}\sum_{v,h} e^{-E(v,h)}\frac{\partial E(v,h)}{\partial\theta} \\
= & \quad \textstyle\sum_h P(v|h)\frac{\partial E(v,h)}{\partial\theta} - \sum_{v,h} P(v,h)\frac{\partial E(v,h)}{\partial\theta} \\
= & \quad E\left[\frac{\partial E(v,h)}{\partial\theta}\middle| v\right] - E\left[\frac{\partial E(v,h)}{\partial\theta}\right].
\end{aligned} \tag{2.16}$$

The final line in equation 2.16 indicates that the parameter update while training RBMs is the difference between the *negative phase* and the *positive phase* of the process. The relationship can be expressed as such, with $E_F(v)$ representing the free energy associated with v, and $\tilde{v}$ representing the reconstructed input of v

from the sampled hidden input.

$$\Delta\theta = \alpha\left(\frac{\partial E_F(v)}{\partial\theta} - \frac{\partial E_F(\tilde{v})}{\partial\theta}\right). \tag{2.17}$$

Given a learning rate α, the parameters W, b_h, and b_v can be updated by the following rules. Note that v is the input vector, which is used to sample the hidden layer h. These two make up the negative phase of contrastive divergence, while $\tilde{x}$ (obtained by sampling h), and $\tilde{h}$ (obtained by sampling $\tilde{x}$) constitute the positive phase. The update rules are defined as:

$$\begin{aligned} W &\longleftarrow W + \alpha(hx^T - \quad \tilde{h}\tilde{x}^T) \\ b_v &\longleftarrow b_v + \alpha(x - \tilde{x}) \\ b_h &\longleftarrow b_h + \alpha(h - \tilde{h}) \end{aligned} \tag{2.18}$$

2.4.4 Deep Belief Network

Using knowledge of SDAs after the process of explaining how dAs work, this section can be cut very short by stating: *"DBNs are to RBMs as SDAs are to dAs"*. The training of the dA and RBM vary greatly as explained in previous sections, however once the weight matricies and bias vectors have been initialized, preprocessing using an n-layered neural network can be accomplished by:

$$Y = W_n(W_{n-1}...(W_1(W_0x + b_{h0}) + b_{h1})...b_{hn-1}) + b_{hn}. \tag{2.19}$$

In my implementation of the DBN, the output Y from the DBN was then given to a logistic regression machine, and classification was completed. The reason for using more than one hidden layer may seem odd, given that one hidden layer of nodes has been proven to be able to express everything that a neural network with a large enough number of nodes can [19]. The number of nodes required in the single hidden layer (called a *perceptron*), and the calculation time and space complexity of the machine is shown to be exponential with respect to the size of the hidden layers [20]. As a real world analogy, the code used to write the DBN classifier of Equation 2.19 consists of 75 lines of python code (using libraries such as numpy and cpickle). If I were not using libraries (the parallel being using less layers in the DBN), the python code would likely be several hundreds if not thousands of lines long. As we think about using less and less levels of abstraction (such as using C not python, or using x86 assembler not C), we can think of the number of lines of code becoming substantially larger. Similarly, we can think of using a single hidden layer perceptron needing billions of hidden nodes, and taking exponentially longer than a deep neural network would.

2.5 Ensemble Methods

A very simple method of ensemble voting was employed for usage in parts of the experiment. Because it is known that partial seizures may only affect a small part of the surface of the brain [21], results can be improved when each individual channel can be classified as having a seizure or not, and the channel votes are combined to produce a final, definitive decision for whether or not a second of data contains a seizure. It is similar to using a majority vote, but in many cases instead of needing $> 50\%$ of the votes for a decision, the ensemble needs $> 15\%$. This value was obtained this thesis and will be elaborated on in Chapter 4 while discussing results.

2.6 Related Work

This study builds upon previous studies in the area of seizure detection, deep belief networks, and time series analysis of high resolution medical data.

In a study by Wulsin [4], deep belief networks were also used for analysis of data obtained from an EEG. The feature set that we chose was a subset of a larger set of features used in the Wulsin study, however this study attempted to classify anomalous EEG features such as *Generalized periodic epileptiform discharges* (GPED), *Periodic lateralized epileptiform discharges* (PLED), or eye blinks as opposed to seizure detection.

A study by Shoeb and Guttag used the same dataset of seizures from the physionet CHB-MIT database of patients that were being monitored by high resolution EEGs after being withdrawn from anti-seizure medications [7]. Although using the same dataset, the Shoeb study extracted a different feature set, and used a support vector machine as the binary classifier, as opposed to a deep belief network. Furthermore, in the Shoeb and Guttag study the EEG data was not fragmented, and statistics were kept on not only the accuracy of seizures detected, but the amount of time that was taken to detect the seizures by the support vector machine.

A final study by Oates et al. [22] explored traumatic brain injury outcomes.

The Oates study investigated time series of high resolution medical data as well, however the data in this study was pulse rate and SpO_2 levels. The study used a Bag of Patterns approach to pre-process data to be used in 1NN clustering to clasify early outcome predictions of patients with traumatic brain injuries.

Chapter 3: Methods And Data

This chapter will explain the database used for this study, as well as the way in which data was prepared.

3.1 MIT Seizure Database

The database used for the experiments is the CHB-MIT scalp database, which is freely available at http://physionet.org/physiobank/database/chbmit/. The data was gathered at the Children's Hospital of Boston, on both males and females aged 1.5-23. All of the participants in the study are patients who suffer from epileptic seizures that had been removed from anti-seizure medications, and were being observed in a hospital setting to judge candidacy for a neurosurgical intervention to prevent the seizures. The types of seizures are not explicitly stated, however all of the seizures are epileptic; there are no psychological seizures present that are undetectable from an EEG. Many of the seizures were partial seizures, and occurred in different areas of the brain. The data provided is in the EDF format, however to be used for machine learning is was first converted to CSV format.

3.2 Experiment Design

Four primary experiments were conducted on the CHB database. The four experiments consisted of raw data classification and featurized data classification on single and multi patient training corpora. The commonality of all of the experiments was the design of the proportional size and purposes of the training, validation, and testing sets. The training set (approximately 5/7 of the data) was the only set that was used for direct adjustment of the weight matrices and bias vectors in the unsupervised learning phase; meaning that the validation and testing data would not be used to obtain gradients or adjust the vector values used in the model. The validation set (approximately 1/7 of the data) was only used to determine model accuracy. The model selected was the one with the smallest error on the validation set, regardless of the accuracy on the test set. The test set (approximately 1/7 of the data) was used as a metric for success of the model. As previously mentioned, the primary metric used is the F_1 measure. If we consider a second where a seizure *is* occurring as a positive instance (P_i), a second where a seizure *is not* occurring as a negative instance (N_i), and false positives and false negatives as F_p and F_n, we can define the precision and recall of the experiment as:

$$\text{precision} = \frac{TruePositive}{OutcomePositive} = \frac{P_i}{P_i + F_p} \tag{3.1}$$

and,

$$\text{recall} = \frac{TruePositive}{CoditionPositive} = \frac{P_i}{P_i + F_n} \tag{3.2}$$

from these we can calculate the F_1 measure,

$$F_1 = 2 \times \left(\frac{precision \times recall}{precision + recall} \right) \tag{3.3}$$

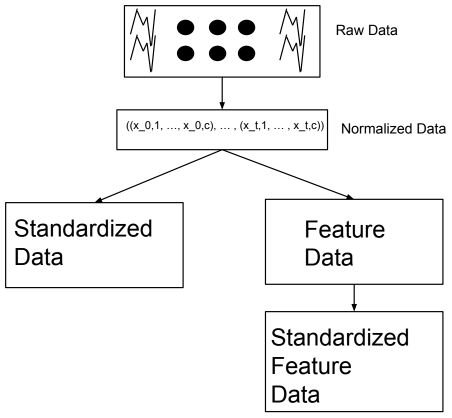
3.2.1 Raw Data Preparation

Raw data classification is appealing because of the simplicity of processing since it is not necessary to extract features from the data. The 23 channels used as input for the EEG were all treated as different distributions of data, so 23 distinct means ($\bar{x}$) and standard deviations(σ) were assigned to the channels. Data was then normalized with $\bar{x} = 0$, and $\sigma = 1$, and the top and bottom 2.5% of data points were truncated, such that $\forall c_i \in |C|$, $x_j \in [-2, 2] \forall x_j \in c_i$. Finally, all of the data x which was now bounded between -2 and 2 was standardized between 0 and 1. For each second of EEG data, a total of 256 hz $\times 23$ channels $= 5888$ points of data were obtained. To test effectiveness of ensemble voting, instead of an input vector of size 5888, each channel was given to the neural networks individually, with input vectors of size 256. The raw data preparation method is the right side of the fork in Figure 3.2.1.

3.2.2 Feature Data Preparation

Preparation of the feature data was a more complicated procedure, which was computationally more intensive. In Figure 3.2.1, after the data was normalized, instead of passing the normalized data to be standardized, the features from Chapter 2 were extracted from the normalized data, and new means and standard deviations

Figure 3.1: Data Preparation Method


Raw Data
((x_0,1, ..., x_0,c), ... , (x_t,1, ... , x_t,c))
Normalized Data
Standardized Data
Feature Data
Standardized Feature Data

for the features of each channel were standardized with $\bar{x} = 0$, and $\sigma = 1$, and then normalized between 0 and 1.

The normalization of feature data was critical because the input to an RBM or SdA must be in the range [0,1]. Although for layers past the first these are probability values, there are not specific layers in the architectures for whether they are input layer or pure hidden, so by normalizing the algorithm input we can use the raw data as if it were a probability (even though it is not). For each second of EEG data, a total of 9 features $\times$ 23 channels = 207 points of data were obtained. Ensemble methods were not tested when feature data was used, so the neural networks had an input size of 207.

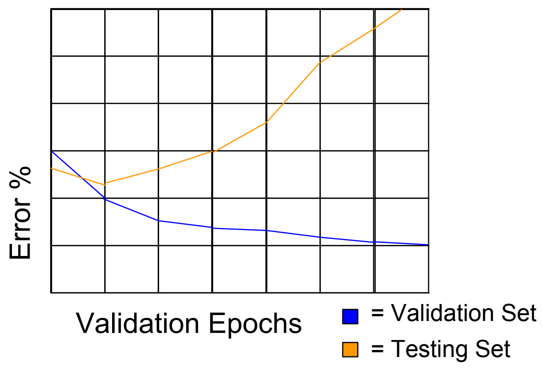
3.2.3 Single Patient Training

For the first type of training and testing setting used, a single patient was trained on, and the same patient was used for validation and testing. There was no intersection between the training set, validation set, and testing set. The purpose of single patient training in this experiment was to establish a benchmark to ensure that the data would be able to yield reasonable results for the types of data presented. The single patient training experiment performed with a greater level of success than the leave one out patient training. To ensure greater results were obtained by superior algorithms as opposed to a luckier breakdown of data between training, validation, and testing groups the same seconds of EEG data were used in single patient training for both raw data input and feature data input.

3.2.4 Leave One Out Patient Training

Of the patients in the CHB database, 9 of them were chosen for deep learning method experimentation; the patients were selected that had channels that were the same as other patients. Patients that did not have all of the standard scalp electrodes, or additional vital signs such as pulse, were not used. *Leave one out* (LOO) patient training consisted of creating a group of 8 patients that would make up the training and validation sets, and a lone patient who would be the testing set. Because of the structure of the experiment, this meant that the test patient had no influence on the values of the weight matricies, bias vector, or parameter selection. This experiment most accurately demonstrates a practical application for seizure detection where no data regarding the patient being tested is given. One of the major advantages of this method is that around 8 times as much data was used for training and validation purposes, and the more data that is available for training, the better the algorithm will be able to learn. This is a slight oversimplification though, because the LOO study yielded poorer results in general than the single patient training did. An amendment to the previous sentence that would be more accurate would be “the more data that is available for training, the greater the algorithm will be able to learn **given the accuracy is measured on data obtained from the same distribution as was trained on.**”

Figure 3.2: Validation and Testing on different distributions



Although all of the patients have been withdrawn from seizure medications, they are of varying ages, and genders. A more likely reason for why the LOO study did worse than the single patient study was because of the variance in types of seizures, and the unique neural pattern of all persons [23]. Since the group of patients that were used for validation of the best model was not taken from the same distribution as the testing data, testing error rates were sometimes inversely related to validation accuracy as shown in Figure 3.2.4. The increased accuracy on the validation set does not represent increased testing set accuracy, similar to how studying very hard for an art history test would not increase performance on a physics test.

Chapter 4: Experiments And Results

Once training and validation optimization functionality was coded, the remaining task of classification was trivial in cases where ensemble methods were not used, as was the case when all channels were given as input to the neural network simultaneously when data was featurized. From here, I will use the term *seizure threshold* (τ) to refer to the number of channels that must indicate a seizure for the second to be classified as a seizure in ensemble methods. In the first of the ensemble optimizations, since the validation and testing set were drawn from the same distribution, the seizure threshold could be chosen in such a way as to maximize the total F_1 measure. In the case of LOO training we used data from the single patient training where the number of channels indicating a seizure that was needed for there to be a seizure ranged from 1 to 6, and the median value of the seizure thresholds was used ($\tau = 4$). In the remainder of this chapter, I will enumerate the four experiments run, and discuss their results and significance to the problem of seizure classification. Afterwards, I will highlight advantages of deep architectures in learning abstracted data in the LOO patient training cases. Using data from the experiments, I will then give a brief quantitative summary of the advantages of using multiple channels of data in seizure detection. Finally, I will discuss some of the shortcomings of this method, with the example of the lower test results of the ensemble method in LOO patient training. It is important to bear in mind while reading the results that the feature data is not the same input as the raw data. One second of feature data is expressed in 207 data points, while one second of raw data is expressed in 5888 data points; furthermore the raw data was given to the SdA, and the feature data was given to the DBN. Throughout this chapter, I will rarely directly compare feature data results with raw data results, and I advise against the reader doing so since they are different kinds of data.

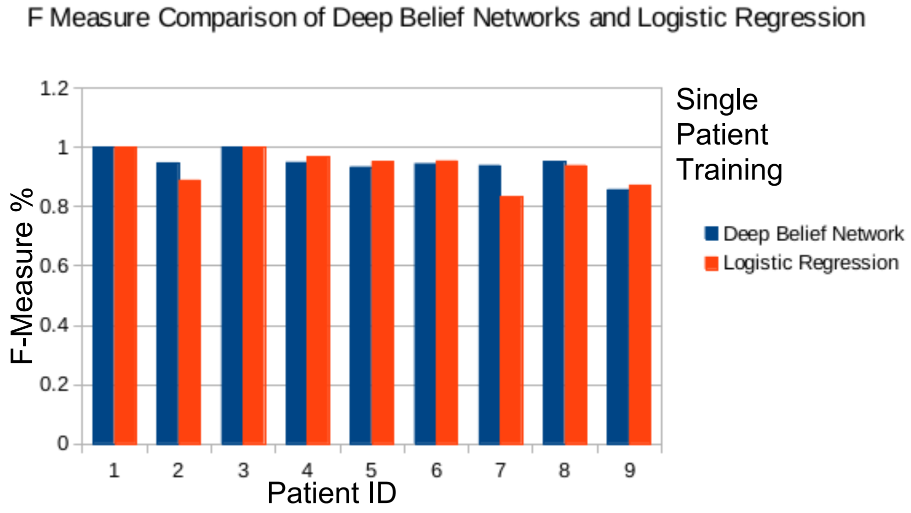
In all four experiments, a two tailed paired sample t test was conducted to determine statistical significance. Because different tests were tested for significance in each experiment, I will explain in the respective sections what is being tested.

4.1 Feature Data Single Patient Training

In the first experiment, feature data was prepared as described in Chapter 3, and each of the 9 single patients were divided to create their own training, validation, and testing sets. The same data was used as input to both the DBN with an LR output layer, and the LR classifier with no DBN preprocessing.

The results of F_1 measure shown in Figure 4.1 speak strongly to the utility of the feature set used. LR classification does not have the flexibility of a polynomial or radial basis kernel used by *Support Vector Machines* (SVM)to make a nonlinear decision surface, but LR is still able to classify seizures very accurately in this environment. The DBN with LR classified slightly better on average, with a mean

Figure 4.1: Featurized Data Single Patient Training

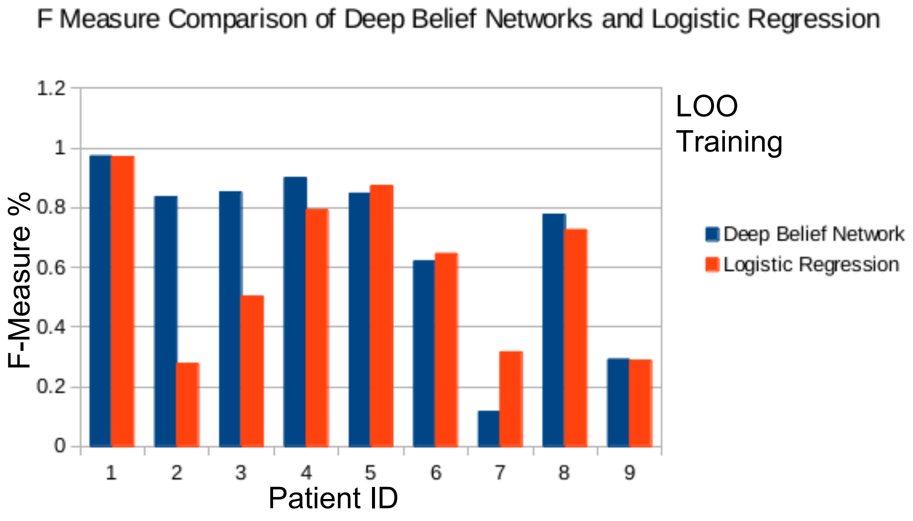


of 94.65, standard deviation 3.97, and a minimum score of 87.71, while the LR classifier had a mean of 93.37, standard deviation 5.42, and a minimum value of 83.3. In this control like group, a significance test was conducted to determine if the predictive properties of a DBN are different from those of an SdA. This hypothesis is only supported with a confidence level of $\alpha = .381$, so if we were to reject the hypothesis that the two identifying properties of the two algorithms were the same, there would be a 38% chance we did so incorrectly.

4.2 Feature Data Leave One Out Patient Training

For the second part of the feature data experiment, the data was prepared with features as it was in the single training instance, but instead of only using one patient for the training group, all of the patients were used except for the patient being tested. Once again, the data was given to a DBN with a LR classifier, and to a LR classifier with no DBN preprocessing.

Figure 4.2: Featurized Data Leave One Out Patient Training



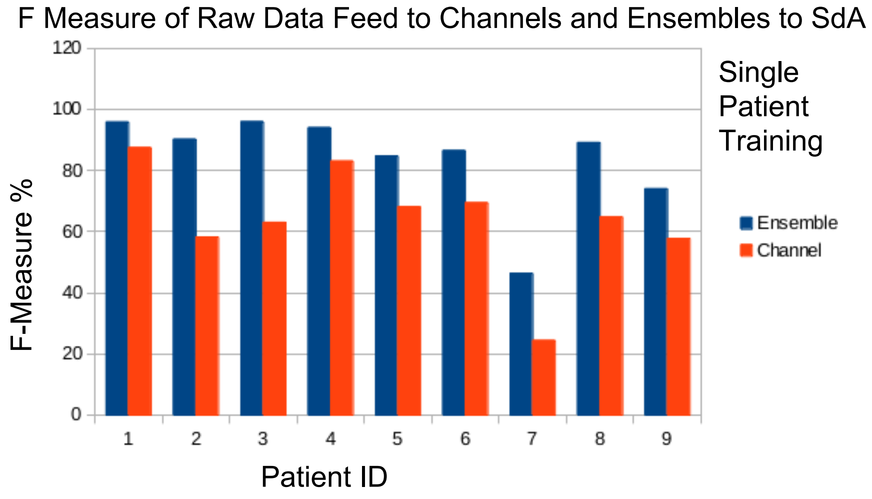
The results of F_1 for Leave One Out patient training are shown in Figure 4.2. The difference between LR classification results with and without DBN preprocessing are more pronounced in this experiment; the mean F_1 measure for DBN (69.03) is almost 10% higher than the mean F_1 measure for LR (59.89). Furthermore, the median difference is almost 20% greater (83.65 for DBN, and 64.49 for LR), and this difference is larger because of an outlying patient for DBN (Patient 7 had an abysmal DBN F_1 measure of 11.63, while the LR had a score of 31.61). More remarks on the expressive power of DBNs will be discussed in Section 4.5. A two tailed paired sample t test of the data shows that the the predictive properties of a SdA are not the same as the predictive properties of a DBN with ($p = .263$). The low confidence level is based on the enormous standard deviations of both of the data sets. Although there is more confidence in the difference between the DBN and LR algorithms in the LOO experiments than the Single Patient experiments, it is still far from definitive. Comparing between Figure 4.1 and Figure 4.2 we see an example of how seizures in the dataset manifest themselves in different parts of the brain. If we look at patient 7 in Figure 4.1 we see that his f measure classification result for both DBNs and LR are not very far from the mean, however in Figure 4.2, patient 7 has a very low score for both algorithms. Although this patient could be classified accurately when their own data was used for training, other patients data was not very helpful in determining the type of seizure.

4.3 Raw Data Single Patient Training

In the raw data experiment, first the case of using a single patient for training and testing purposes was done, as was the case for feature data. LR Classification was not used without DBN preprocessing in this experiment, because of the inability of LR to learn on raw data sets with no preprocessing. Although LR is a powerful algorithm as shown in earlier experiments, it lacks capabilities of abstraction as a deep architecture does. For example, if LR detected an EEG artifact indicating a seizure in the first part of the second, it would be unable to use that knowledge to know that the same exact artifact in the later part of a second would indicate a seizure.

In this study, ensemble methods were employed, and to capitalize on the effectiveness of ensemble voting, single channel classification is plotted against ensemble methods. The ensemble method had a mean F_1 value of 84.04, with a standard deviation of 14.81, while the individual channels have a mean F_1 value of 58.86, with a standard deviation of 15.82. The difference of F_1 measure between these

Figure 4.3: Raw Data Single Patient Training



mean values is 25.18, which is quite large. The τ value for the ensembles varied between 1 and 6, with a mean of 4.11, and a median and mode of 4. The patient with the lowest F_1 measure of 46.34 for ensemble and 24.42 for individual channels had a seizure threshold value of $\tau = 1$. This was the minimum threshold value for all of the patients in this study (compared to the second lowest τ value of 4 which was the median and mode), and may shed insight into the types of the patient's seizure. Because the EEG is glued on the scalp and is not capable of measuring abnormal electrical activity on other parts of the brain, some types of seizures can be much more difficult to capture via an EEG, and may show minimal electrical disturbances on a small number of channels [24]. In this test statistical significance was calculated between the F_1 measure of a single channel, and the F_1 measure of the ensemble of channels, $p = .0001$. Although this number is at first glance very impressive, we must keep in mind that an average seizure could only be observed in 4 out of the 23 channels. The dominance of the ensemble justifies multiple channels, as will be elaborated on further in Section 4.6.

4.4 Raw Data Leave One Out Patient Training

In the second part of the raw data experimentation, one patient was excluded from the test corpus. Two sub experiments were run; one in which optimization on the τ value was performed on the validation set, and one in which the optimization was performed on the testing set. The latter is frowned upon in the machine learning community, and for good reason. If we are optimizing on a test corpus, we can reach significantly higher levels of performance without gaining any advantage in classifying new unseen data. The test set optimization was performed to demonstrate the difficulty of optimizing from a validation corpus different from the testing corpus.

Figure 4.4: Raw Data Leave One Out Patient Training

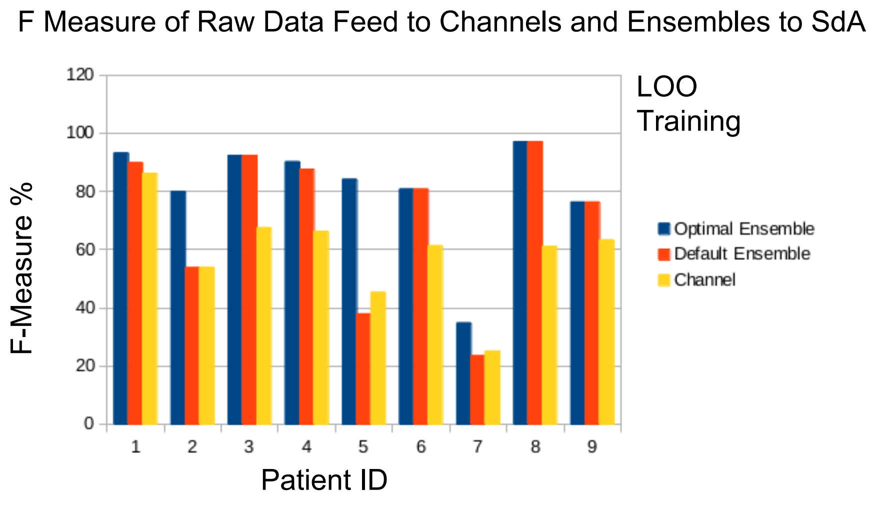


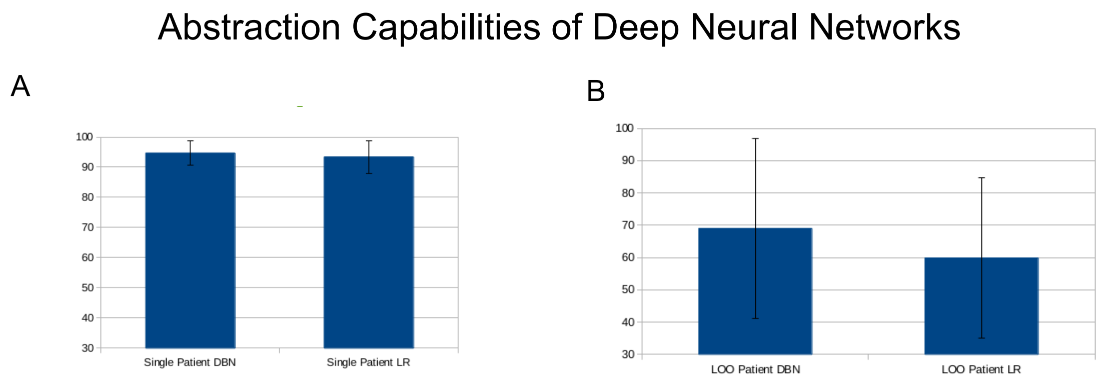
Figure 4.4 differs from others because of the added averages from the test set optimization. We will not discuss this strange dataset until section 4.7, and for the remainder of the section, the ensemble group will refer to the lower accuracy model (*default model* in Figure 4.4) obtained via validation set optimization. The mean

value for the ensemble was 71.03, with a large standard deviation of 24.79, and a minimum value of 23.53. The raw channels classified had a mean of 58.86, with standard deviation 15.82, and a minimum value of 25.14. This experiment highlights the abstraction capabilities of deep architectures, as well as displays the significance of additional channels. These strengths, as well as the weakness of optimization on a different data distribution is discussed in the following sections. Statistical significance testing was done between the raw channel data, and the default value of 4 channels. The predictive capabilities of a single channel was not equal to the predictive capabilities of the default ensemble with a confidence level of $\alpha = .035$.

4.5 Abstraction Capabilities

A strength of deep architectures lies in their ability to abstract information from the input, and build a model that can classify similar types of input data more accurately not only in the arena of robotic vision, but for tasks as difficult as natural language processing [25].

Figure 4.5: Data Abstraction



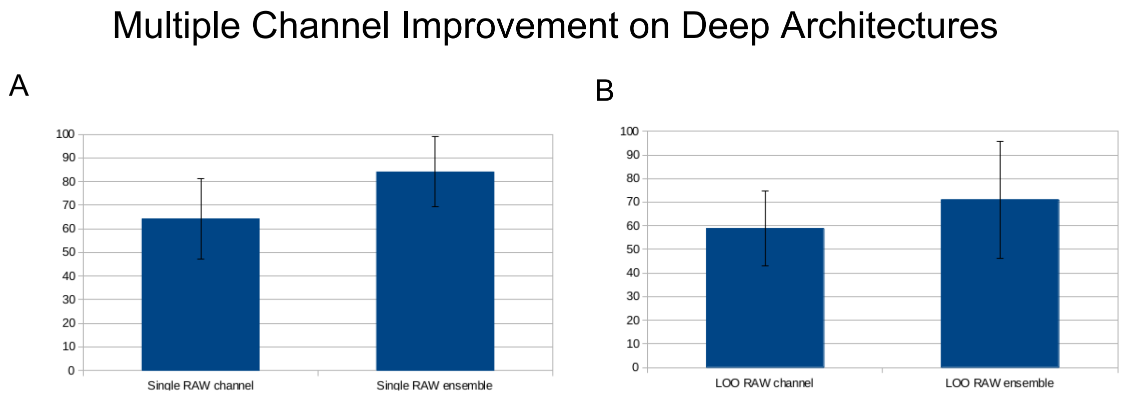
Shown in Figure 4.5 is the difference between the F_1 measures between the

DBN with LR compared to the LR classifier. In Figure 4.5 (A), we see that both of the classifiers performed very well, with a mean F_1 measure of over 90 for both. The largest difference between the two classifiers is that the LR with no DBN pre-processing had a larger standard deviation. In the more difficult machine learning problem of classifying seizures for a patient by training on other patients, we see in Figure 4.5 (B) that the mean F_1 measure for DBNs is about 10% greater than for LR with no DBN.

4.6 Advantages of Multiple Channels

The addition of more data in machine learning can allow for better classification, as it's no surprise that *more relevant data is good.* In the seizure classification problem, extra data is not only helpful, but may be a necessity since many seizures do not manifest themselves globally across the brain, and detection with few channels of an EEG may be impossible [24].

Figure 4.6: Accuracy increases with multiple channels



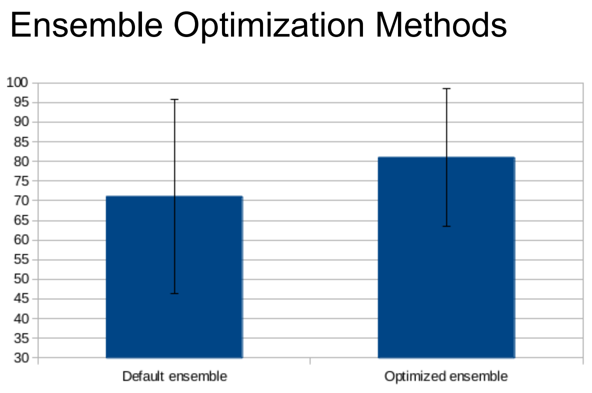
In all of the 9 patients tested in the single patient experiment, the ensemble classification of the 23 channels had a higher F_1 measure, with a mean difference of 20.0 points (84.04 for ensemble, and 64.04 for single channel). An important note for the single channel score of 64.04 is that the seemingly low score may be impacted by the fact that a seizure was not present at that electrode. If a frontal lobe seizure was occurring, then the T5-P3 electrode in the back of the head would likely not detect any irregularities, as the electrical disturbance may only be observable for a few square centimetres on the surface of the brain [24]. The difference between single channel and ensemble F_1 measure for LOO patient training is less substantial at a mean difference of 12.17. The difference may be less than the single patient experiment because of the classification difficulty of improperly optimized ensembles, as is explained in more detail in section 4.7.

4.7 Ensemble Limitations

A final important conclusion that can be drawn from the dataset is the difference between the performance of ensemble optimization on the patient corpus not containing the test patient, and the corpus that did contain the test patient.

The differences between using the validation set for the ensemble optimization and the testing set in the ensemble optimization are shown in Figure 4.7. The difference of F_1 measure mean between the two sets is 9.93, which is so significant because the same data was run through the same weight matrices and same bias vectors, the only difference was the number of channels that would have to indicate a seizure for the second to be labeled a seizure. The predictive capabilities of the test

Figure 4.7: Ensemble Optimization Methods



set optimized ensemble is only statistically different from those of a default ensemble at a confidence level of $p = .1012$. Although the difference is not statistically significant, we know that the training set ensemble can never be more efficient than the test set optimized ensemble.

In a practical setting, if doctors are aware of what types of seizures the patient suffered from, they may be able to customize the τ value, as this patient specific knowledge can be useful in interpreting EEG signals [24]. A further extension of this could be applied to the training set; if only patients with the same types of seizures on the same parts of the brain were trained on then seizure classification could be even more accurate.

Chapter 5: Conclusion And Discussion

This study explored a variety of the new deep learning methods used on multichannel time series EEG data. Three important things to remember about deep neural networks are:

- The abstraction capabilities of deep neural networks is observable even when a very strong feature set is given to the algorithm as input. The DBN had a slight edge against LR on feature data.

- The more channels that are available for the algorithms use, the more likely there will be a noticeable improvement. Some parts of the brain are hard to observe, and having a higher density of scalp electrodes nearby to detect electrical changes will

- Patients suffer from various different seizures which manifest themselves in different ways observable by an EEG.

Using these facts, experiments were run on a variety of seizure patients attempting to classify the seconds of the labeled EEG data as seizure or nonseizure. In the first experiment using data abstracted from the EEG to form features, we observe that the difference between the DBNs ability to detect seizures is more noticeable in the setting where the patient used to construct the model in the training phase is the same patient being tested on. In the second set of experiments involving the classification of raw data, we see how the addition of multiple channels increases performance, and how when the individual channels are polled for a decisive answer for the entire second, the classification accuracy increases. Finally, this study demonstrates some of the disadvantages of not having patient specific information to tune the networks for what types of seizure is occurring.

The thesis lends itself to future work in the areas of seizure specific classification. If instead of training on all patients in the corpus we were to train on only patients with the same types of seizure we would likely achieve a much higher test accuracy. Another way of improving accuracy that is not adding more EEG channels is the addition of other vital sign and body monitors such as pulse rate, oxygen saturation, even accelerometers attached to limbs. This could be especially helpful on seizures where the electrical disturbance is virtually undetectable on the scalp.

A final future work in this area is brain computer interfacing and telepathy. In my first experiences working with EEGs, I received a commercial EEG from *Emotiv*, and experimented with brainwaves and what made EEGs produce different electrical readings. I found that more often than not it would be able to differentiate between two binary thoughts (thinking of red vs. thinking of blue). This would be an exciting application of the ability of deep architectures in processing and classifying EEG signals.

Chapter A: Machine Learning Parameters

Logistic Regression

Part	Parameter	Value
LR	Input	207
LR	Output	2
LR	Learning Rate (α)	.13

Deep Belief Network

Part	Parameter	Value
DBN	Hidden Layers	[500,500]
DBN	Pretrain Learning Rate (α_1)	.001
DBN	Pretrain Epochs	36
DBN	Gibbs Steps	1
LR	Input	500
LR	Output	2
LR	Fine Tune Learning Rate (α_2)	.1

Stacked Denoising Autoencoder

Part	Parameter	Value
SdA	Hidden Layers	[1000,1000]
SdA	Hidden Layer Corruption (β)	[.08,.08]
SdA	Pretrain Learning Rate (α_1)	.001
SdA	Pretrain Epochs	49
LR	Input	1000
LR	Output	2
LR	Fine Tune Learning Rate (α_2)	.1

Bibliography

[1] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[2] N. Sivasankari and K. Thanushkodi. Epileptic seizure detection on eeg signal using statistical signal processing and neural networks. In *Proceedings of the 1st WSEAS international conference on Sensors and signals*, SENSIG'08, pages 98–102, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[3] H.R. Mohseni, A. Maghsoudi, and M.B. Shamsollahi. Seizure detection in eeg signals: A comparison of different approaches. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, volume Supplement, pages 6724–6727, 2006.

[4] D F Wulsin, J R Gupta, R Mani, J A Blanco, and B Litt. Modeling electroencephalography waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement. *Journal of Neural Engineering*, 8(3):036015, 2011.

[5] I. Arel, D.C. Rose, and T.P. Karnowski. Deep machine learning - a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18, 2010.

[6] Y lan Boureau and Yann Lecun. Sparse feature learning for deep belief networks.

[7] Ali Shoeb. *Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment*. PhD thesis, Massechusetts Institute of Technology, Cambridge, September 2009.

[8] Wanli Min and Gang Luo. Medical applications of eeg wave classification. *CHANCE*, 22(4):14–20, 2009.

[9] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[10] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[11] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

[13] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[14] M.J. Embrechts, B.J. Hargis, and J.D. Linton. Augmented efficient backprop for backpropagation learning in deep autoassociative neural networks. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–6, 2010.

[15] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. Technical Report arXiv:1206.5533, U. Montreal, Lecture Notes in Computer Science Volume 7700, Neural Networks: Tricks of the Trade Second Edition, Editors: Grégoire Montavon, Geneviève B. Orr, Klaus-Robert Müller, 2012.

[16] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM.

[17] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 153–160. MIT Press, 2007.

[18] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[19] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.*, 10:1–40, June 2009.

[20] Hugo Larochelle, Yoshua Bengio, Jrme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40, 2009.

[21] Andrew B. Gardner, Abba M. Krieger, George Vachtsevanos, and Brian Litt. One-class novelty detection for seizure analysis from intracranial eeg. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:1025–1044, 2006.

[22] T. Oates, C.F. Mackenzie, L.G. Stansbury, B. Aarabi, D.M. Stein, and P.F. Hu. Predicting patient outcomes from a few hours of high resolution vital signs data. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 192–197, 2012.

[23] Heather C. Mefford, Hiltrud Muhle, Philipp Ostertag, Sarah von Spiczak, Karen Buysse, Carl Baker, Andre Franke, Alain Malafosse, Pierre Genton, Pierre Thomas, Christina A. Gurnett, Stefan Schreiber, Alexander G. Bassuk, Michel Guipponi, Ulrich Stephani, Ingo Helbig, and Evan E. Eichler. Genome-wide copy number variation in epilepsy: Novel susceptibility loci in idiopathic generalized and focal epilepsies. *PLoS Genet*, 6(5):e1000962, 05 2010.

[24] S J M Smith. Eeg in the diagnosis, classification, and management of patients with epilepsy. *Journal of Neurology, Neurosurgery Psychiatry*, 76(suppl 2):ii2–ii7, 2005.

[25] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 582–590, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.