FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Training Autoencoders for State Estimation in Smart Grids

Rui Miguel Machado Oliveira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Vladimiro Henrique Barrosa Pinto de Miranda

Second Supervisor: Ricardo Jorge Gomes de Sousa Bento Bessa

July 25, 2017

Resumo

Desde a sua implementação massiva nos centros de controlo dos sistemas elétricos de energia, o estimador de estado apresenta-se como uma ferramenta de capital importância uma vez que permite a obtenção de um ponto de funcionamento do sistema com elevada precisão. Com efeito, uma das etapas essenciais à definição desse ponto de funcionamento é a determinação da topologia, isto é, o estado aberto ou fechado dos dispositivos de corte e seccionamento da rede em questão.

Assim, uma miríade de metodologias foi proposta com o firme intuito de endereçar o problema da identificação de topologia. Os primeiros passos foram dados em meados dos anos 80 sendo apresentadas abordagens baseadas em métodos puramente matemáticos e na análise dos seus resultados para a deteção de eventuais erros de topologia. Todavia, através da crescente popularidade e dos avanços nas áreas da inteligência artificial e da ciência da computação, foram desenvolvidos modelos baseados em redes neuronais, evidenciando resultados convincentes com pesos computacionais mais reduzidos e não apresentando os problemas de convergência dos métodos estritamente analíticos.

Tendo isto em consideração, nesta tese é ilustrada uma nova perspetiva da aplicação de redes neuronais, *deep learning* e de aprendizagem baseada em teoria da informação (ITL) ao problema da determinação de topologia e, em última instância, à estimação de estados. Através da conjunção destes conceitos, pretende-se identificar a topologia de um dado sistema elétrico de energia recorrendo ao reconhecimento dos padrões das medidas analógicas providenciadas, apontando, ao mesmo tempo, metodologias alternativas ao nível do treino e arquitetura de redes neuronais bem como de ferramentas computacionais que conduzam a uma maior eficiência desse processo.

Particularizando, serão exploradas duas filosofias diferentes com elementos de treino supervisionado e não supervisionado e com arquiteturas típicas de autoencoders e redes neuronais *feed-forward* aplicadas à descoberta da topologia de um sistema IEEE RTS 24. Para além do propósito de contrastar a eficácia de cada um dos métodos, serão igualmente demonstradas comparações no que concerne o peso e rapidez computacional através da utilização de CPU e GPU, atestando as potencialidades desta última técnica em aplicações concretas nos sistemas elétricos de energia.

Palavras-chave: Estimação de estados, topologia de um sistema eléctrico de energia, teoria da informação, redes neuronais, autoencoder, *deep learning*

Abstract

Since its massive deploy in control centres, the state estimator is regarded as a tool of critical importance due to its capability of determining a system's operating point with remarkable precision. Indeed, one of the essential stages to obtain that operating point is identifying the system topology, i.e., the open or closed status of the grid's switches and breakers.

As a result, a considerable amount of methodologies was proposed with the purpose of detecting the correct topology. First efforts related to this subject date back to the 80's and preconized the use of strict mathematical equations and, subsequently, the analysis of their results to evaluate possible topology errors. However, other techniques showed the advantages of the application of artificial neural networks to this particular problem; among them, figure the reduced computational burden and immunity to convergence problems of mathematical, analytic methods.

Hence, this thesis presents a novel perspective of the application of neural networks, deep learning and Information Theoretic Learning (ITL) to the problem of topology determination and, ultimately, to state estimation. Blending these concepts, the main objective is to predict the topology of a given power system based on pattern recognition of analog measurements, pointing, at the same time, alternative approaches on neural networks' training and architecture as well as computation tools capable of introducing enhanced efficiency to that process.

With more detail, two different applications with elements of supervised and unsupervised training as well as with autoencoder architectures and typical feedforward neural networks will be explored to discover the status of a breaker device of an IEEE RTS 24 test case. Apart from comparisons between their efficacy, other major point will be the contrasts between running times achieved by CPU and GPU computation, showing that the large spectrum of application of this last technique comprises also power systems.

Index Terms: State estimation, power system topology, Information Theoretic Learning, neural networks, autoencoder, deep learning

Acknowledgments

First of all, I would like to thank my thesis supervisor Professor Vladimiro Miranda. On one hand, by being the intellectually challenging professor and person that he is. On the other hand, I am deeply grateful to him for giving me the encouragement and the opportunity of working in such interesting and innovative fields. I also want to address a great acknowledgement to my co-supervisor, Dr. Ricardo Bessa, for his availability and openness to my natural doubts as a master's degree student.

I also want to thank Pedro Cardoso for his marvelous friendship and work partnership in all these college years.

Additionally, I want to thank to my family, in particular to my parents and brother, for their continuous support. They made me who I am today through their values, lessons and love and I am forever thankful to them for that.

Finally, I want to express my profound gratitude to my beloved girlfriend, Carlota. Thanks for being the special person that you are and for being the balm to my soul.

Rui Miguel Machado Oliveira

"Pois não é evidente, Galileo?
Quem acredita que um penedo caia
com a mesma rapidez que um botão de camisa ou que um seixo da praia? "

António Gedeão

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

AE	Autoencoder
ANN	Artificial Neural Network
CPU	Central Processing Unit
EMS	Energy Management System
GPU	Graphic Processing Unit
GSE	Generalized State Estimation
IEEE	Institute of Electrical and Electronics Engineers
ITL	Information Theoretic Learning
MEC	Maximum Entropy Criterion
MLP	Multilayer Perceptron
MQMIC	Maximum Quadratic Mutual Information Criterion
MSE	Mean Squared Error
OPF	Optimal Power Flow
PCA	Principal Component Analysis
RTS	Reliability Test System
SCADA	Supervisory Control and Data Acquisition
SE	State Estimation
SGD	Stochastic Gradient Descent
WLS	Weighted Least Squares

Chapter 1

Introduction

In this chapter, a brief overview will be performed regarding the main topics related to this work, specifically the importance and usefulness of machine learning to state estimation and the exploration of information-theoretic principles in power systems. Subsequently, the purpose of this thesis is given, stating the main goals of the developed work. At last, the organization of this thesis is presented.

1.1 Motivation

State estimation in power systems corresponds to the process of estimate the values of electrical variables of a power system. The traditional archetype of state estimation is characterized by mathematical models which try to estimate a plausible operating point given the set of analog measurements.

One of the assumptions for this classical state estimation is that the topology of the system - defined by open or closed state of the breaker and switching devices - is already known. However, topology errors may occur and greatly jeopardize the process of state estimation due to its binary nature.

In order to face this problem, several strategies were proposed, since branch power flow checking to linear programming formulations. Nonetheless, along with the exponential popularity growth of computer science, machine learning and, ultimately, artificial neural networks, there was a rising adoption of techniques belonging to those paradigms in power systems, particularly in topology estimation. Therefore, a deviation from the classic, rigid models was done in order to grant a more automatized, fast way of dealing with proper idiosyncrasies of this field.

Through the exhaustive analysis of underlying training procedures to such artificial neural networks solutions, it can be observed that those procedures are merely based on criteria that do not consider the input data characteristics; i.e., the objective is only to maximize the success rate of the model. Despite the existence of interesting results in some approaches, this *modus operandi* is not totally satisfactory.

In other words, input data characteristics in topology estimation can be perceived as information about the manifold that contains such input vector. In a first glance, the definition of information can be rather vague and not very helpful. Notwithstanding, Shannon's mathematically derived concepts - entropy and mutual information - may be adopted, giving a much more treatable character to information quantification and leading to an information theoretic learning standard. This way, the importance and the benefit of combining the best of both worlds becomes clear. That is, there is the necessity of the development of an artificial neural network solution that takes into account the distinct nature of the information enclosed by mathematical power flow equations that support state and topology estimation. In this thesis, this challenge is addressed and solutions will be presented regarding a world-wide reference test case.

1.2 Objectives

This work has the clear intention of being a vehicle of achieving a better and different way of performing topology estimation, taking advantage of advanced paradigms such as artificial neural networks, machine learning and information theory. More properly, it is desired to develop a tool which can predict the topology of a given system based on pattern recognition of analog measurements.

From another perspective, it is also desired to show the potential and usefulness of using information theory descriptors in ANN training and in state estimation. As opposed to conventional, variance-centered methods, information theory methods rely on notions of entropy and mutual information. Those same descriptors allow us to focus on the most important data trait which is its actual information content, giving the possibility of building more adequate and accurate models, taking into account not a rather simple and rudimentary assumption of prediction error but the real data distribution.

1.3 Original Contributions

The main contributions of this thesis for the state of the art of state and topology estimation are:

- Training of deep autoencoders and deep multilayer perceptrons for local topology estimation with ITL criteria. Use of maximum entropy cost functions.

- Assessment of the usefulness of GPU computing in the reduction of training times of neural network applications in state and topology estimation applications. Overview of GPU and CPU computational gains.

- Analysis of the impact of models' hyperparameters, processes and techniques in breaker status prediction accuracy.

- Comparative evaluation of the performance of two methodologies - Deep AE and Deep MLP - in breaker status identification.

1.4 Thesis organization

This thesis is composed by five chapters. The first one serves as an exposure of the purpose and fundamental aims of this thesis as well as an introduction to some important notions related to new perspectives on state and topology estimation.

The second chapter corresponds to State of the Art. In this chapter, a review about artificial neural networks, autoencoders and their training procedures is performed. Then, an overview to classic state estimation using the Weighted Least Squares is presented, being followed by a comprehensive state of the art about topology estimation. Finally, the connection between autoencoders and its application in state estimation is highlighted, giving a portrait of the existent scientific production.

In Chapter 3, a bridge between the state of the art and the explanation of both training methodologies is firstly performed through the formulation of topology estimation and the explanation of basic concepts of information theory, machine learning and information theoretic learning. Then, a thorough presentation of the inner mechanisms of both methods - Deep AE and Deep MLP - is done, specifying aspects like ANN architecture, weight initialization, optimization algorithms or training procedures.

In Chapter 4, the prediction results of both methodologies are presented for a single breaker example case. Subsequently, two important features are going to be subject of comparison: the prediction accuracy and running times of train and test of Deep AE and Deep MLP. At last, the dominant rationales behind the exposed results are going to be analyzed.

Finally, in chapter 5, the main conclusions will be explicitly depicted and some suggestions for future developments will be given.

Chapter 2

State of the Art

This chapter provides a necessary introduction to fundamental concepts in which this dissertation is related to: autoencoders and its application to the state estimation problem. Each one of the topics enumerated above is depicted in a different section. With this chapter, the author aims to give the reader a clear explanation of the proposed concepts as well to provide a relatively exhaustive literature review, highlighting significant developments in those areas.

2.1 Autoencoders - Concepts and Applications

This section has the objective of reviewing some cornerstone principles of autoencoders. In a first moment, there is a short exposure of the concept of artificial neural network, being followed by a clarification of the idea behind autoencoders and their training procedures.

2.1.1 Artificial Neural Networks - a brief overview

Artificial Neural Networks (**ANN**) can be defined as a computation tool inspired by the biological model of the brain. An ANN can learn to solve a problem based on a given set of examples, gaining, with proper training, a sense of generalization; i.e. use of previously acquired knowledge to new scenarios. This approach may be particularly useful in certain types of problems when there are no rigid algorithms to solve them.

Like their biological counterparts, artificial neural networks possess their fundamental unit, the artificial neuron, which can be seen as the basic processing element of these structures. A neuron has normally several inputs, each one of them affected by a specific weight, and an output, result of the processing of this unit. This input-output mechanism can be represented mathematically by the subsequent expressions:

$$net_{kx} = \sum_{z=1}^{m} w_{kx,pz}\, O_{pz} \qquad\qquad O_{kx} = f_{kx}(net_{kx} - b_{kx}) \tag{2.1}$$

Where:

- net_{kx} is the input signal of neuron x at layer k;
- $w_{kx,pz}$ is the weight associated with the connection between the neuron z in layer p and neuron x at layer k - layer p precedes the layer k;
- O_{pz} is the output signal of neuron z at layer p;
- O_{kx} is the output signal of neuron x at layer k;
- f_{kx} is the activation function of neuron x at layer k - linear, ramp or sigmoid are standard examples of ANN activation functions;
- b_{kx} is the activation threshold of neuron x at layer k;

One of the most popular methods for training ANN's is the backpropagation algorithm combined with gradient descent. The main idea behind this method is minimizing the difference - or cost function - between the output provided by the neural network, y', and the desired output, y, by calculating the gradient of the error in order to the various weights, changing them accordingly. In a synthetic manner, this iterative method can be described by the following steps:

1. Perform a feedforward computation to activate all the layers, to get an output y';
2. Measure the deviation between y and y' - a wide range of metrics may be applied;
3. Backpropagate the error, updating the weights;
4. End the algorithm when the error is smaller than a defined threshold;

Given the need of a target output, backpropagation algorithms are normally associated with supervised learning, even though they make part of training processes of fundamentally unsupervised networks such as autoencoders. Examples of alternative training techniques will be discussed later in more detail from an autoencoder-oriented angle.

ANN's may appear with two types of architectures: feedforward or recurrent. In the first category there is the propagation of information along the direction input-output, not having any kind of feedback process which could introduce the influence of downstream results in upstream processing of the network. On the other hand, in recurrent ANN's there is that possibility, existing more arbitrary topologies; still, stability in outputs and proper training of this type of neural networks are more difficult to obtain.

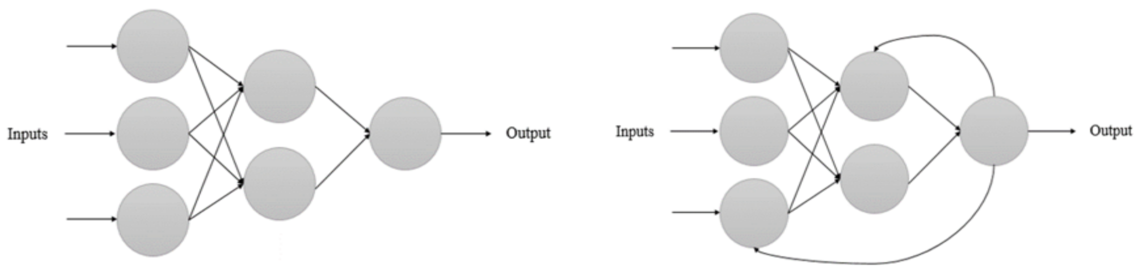

Figure 2.1: Feedforward and recurrent ANN's, respectively.

2.1.2 Autoencoders

Autoencoders (**AE**) are feedforward artificial neural networks trained with the objective of getting an output which mirrors its input with the minimum distortion possible. However, capacity of generalization is mandatory: if the proposed autoencoder only reproduces the given input in its output, then the training procedure has failed the task of learning useful information from the training dataset, or, more simply, there is an overfitting of the AE [1].

This kind of ANN may be viewed as the junction of two components: the first half - encoder - attempts to represent the input X in a space with less dimensions X' through the use of an encoding or compression function f. On the other hand, the second half - decoder - aims to do exactly the opposite, aiming to approximate f^{-1}.

The typical structure of an autoencoder is to have an input layer, an hidden layer and an output layer as depicted in Figure 2.2. A configuration containing more hidden layers is showed throughout specialized literature [2]. Even though a correlation between the number of hidden layers and higher accuracies is reported, that is achieved through a more refined and demanding training procedure.

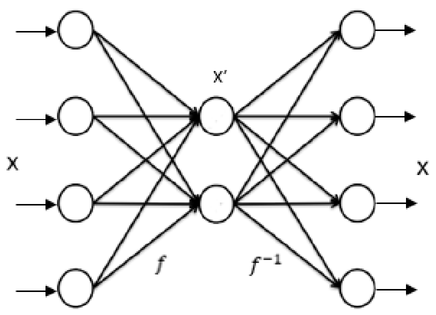

Figure 2.2: Common architecture of an autoencoder.

Still related to its structure, there is no canon or widely-accepted rule in which it is portrayed hints about the optimal arrangement of layers or the number of neurons of an autoencoder. For each situation, there is a trial-and-error analysis to determine those parameters.

Prior to the use of autoencoders in reduced-space representation problems, there was **Principal Component Analysis** (**PCA**). This technique can be described as a statistical procedure which aims to represent a given dataset - in which there is possibly mutually correlated variables - by linearly uncorrelated variables (principal components), performing a dimensionality reduction. In [3], it is concluded that an autoencoder with linear activation functions makes an information compression very much in the way of PCA. In theory, using linear autoencoders would be an advantage because one could split the autoencoder, calculate the first half weights based on a PCA process and using the transposed weights' matrix for the second half, eliminating the need of training. However, PCA is restricted to a linear mapping while autoencoders can extract relevant features using non-linear models, having a greater range of applications.

Taking in consideration their main capabilities, autoencoders were firstly used in the 90's in common tasks for ANN such as image compression, feature extraction and missing data restoration [4–6]. As the diversity of applications grown, some variations of the AE concept were conceived to better tackle supplementary issues such as enhancing the robustness of feature extraction or noisy data. So, in specialized literature, one can easily find terms as denoising autoencoder or stacked denoising autoencoder [7, 8]. As the name suggests, both are trained to provide an output consisting of a corruption-free version of the input; nevertheless, fundamental differences are observed in terms of architecture: a stacked denoising autoencoder is, basically, a chain of denoising autoencoders, forming deeper models. Further references to additional configurations of autoencoders - for example, variational or contractive AE's - can be found in [9, 10].

2.1.3 Training processes of autoencoders

As illustrated above, one can, fundamentally, define training of an ANN as the process of determining the weight values that suit best to the required model, optimizing the cost function with a satisfactory computational burden. Therefore, in this chapter notions such as cost function, training strategy or weight initialization of autoencoders are going to be introduced.

In broad terms, cost function, also called loss or error function, corresponds to the measure of how well the ANN is performing during the training process; i.e. it is the metric used to evaluate the difference between the actual, y^a, and aimed output, y^o. There is a multitude of cost functions and, as a result, one of the fundamental steps to design a machine learning algorithm is to carefully choose the cost function that is the most adequate to the type of problem. Minimization of quadratic cost functions, such as the sum squared error and the mean squared error (MSE), are the most widely applied criteria in neural network algorithms; cross-entropy (CE) and exponential (EXP) cost functions are also often employed. In [11], a comparative study between sum squared error, cross-entropy and exponential cost function is made regarding the stacked autoencoders' training.

$$C_{MSE} = \frac{1}{n}\sum_j (y_j^a - y_j^o)^2 \tag{2.2}$$

$$C_{CE} = -\sum_j [y_j^o \, ln \, y_j^a + (1 - y_j^o) \, ln \, (1 - y_j^a)] \tag{2.3}$$

$$C_{EXP} = \tau \exp\left(\frac{1}{\tau}\sum_{j}(y_j^a - y_j^o)^2\right) \tag{2.4}$$

Despite their usefulness in supervised learning, conventional cost functions are not applicable in unsupervised learning tasks. One example is dimensionality reduction, in which one does not know the data representation in the compressed space [12]. With that in mind, information theoretic learning (ITL) cost functions can be adopted with the purpose of retaining the maximum information possible in the ANN [13, 14]. A considerable amount of criteria can be derived from ITL principles, being notable examples the maximum entropy or maximum quadratic mutual information. A more detailed review of ITL cost functions and their main concepts is going to be carried out in Chapter 3.

On what concerns strategies for training autoencoders, typical backpropagation methods are adequate just as in any other feedforward ANN. Nevertheless, the efficiency of the algorithm is strongly dependent on the number of hidden layers of the autoencoder and activation functions: problems such as vanishing or exploding gradient may constantly occur, resulting in slow convergence or undesired solutions. Consequently, diverse variations of classical backpropagation with distinct optimization approaches, such as Levenberg-Marquardt [15] or conjugate gradient descent [16], as well as other alternatives, for example the resilient backpropagation [17], were initially proposed to address this issue.

Besides the optimization algorithm, one can also point that the importance of the initial weights in the training procedure: if their values are close to the final solution, adequate convergence is achieved faster. This perspective is perceived in [18] in which Hinton *et al.* report the design of a a new learning algorithm for Deep Belief Networks consisting in unsupervised pretraining based on Restricted Boltzmann Machines followed by a regular backpropagation for fine-tuning, also applied to autoencoders in [2]. This algorithm can be shortly described by the following steps:

1. Split the autoencoder in two halves: the encoder and decoder;
2. Submit the encoder to a greedy layer-wise unsupervised pretraining through a stacked Restricted Boltzmann Machine architecture, initializing the encoder weight values;
3. Unfold the model, considering the decoder a "mirrored" encoder;
4. Perform a backpropagation supervised training to all the ANN;

Although this latter strategy has interesting characteristics of rapid convergence and enhanced generalization, it is a rather intricate technique. As a matter of fact, works portraying improvements in optimization methods [19], alternative initialization procedures [20] or architectures without pretraining led to a progressive abandonment of this type of approach. Furthermore, the use of rectified linear units (ReLU) equally contributed to this relative decay of Hinton's view due to their capability of being less affected by gradient problems and for being easier to compute; the application of rectified linear units in deep ANN's is depicted in [21].

Another strategy for training autoencoders was proposed more recently in 2014 by Miranda *et al.* in [12] using a shallow autoencoder configuration - one input layer, one hidden layer and one output layer. More particularly, this technique advances a novel and more computationally efficient way of training autoencoders by training half of the autoencoder in each step, adopting ITL and quadratic cost functions. This way, this algorithm can be depicted as the sequence represented below:

1. Split the autoencoder in two halves: the encoder and decoder;

2. Initialize the encoder weights as if it was PCA, calculating the eigenvectors of the input data;

3. Submit the encoder to a unsupervised training, maximizing the information flow between the input and hidden layer.

4. Fixing the encoder weights, submit the decoder to supervised training, minimizing the error between the input and output through a MSE criterion;

Considering weight initialization, one is faced with a significant number of tendencies. Apart from the pretraining procedure proposed by Hinton *et al.*, straightforward random techniques were developed in [22] and [20] which are, still, regularly adopted. In the first work, Le Cun *et al.* proposes the determination of weights, w_{ij}, following a uniform probability distribution affected by the n_{in}, the size of previous layer. So, for the sigmoid:

$$w_{ij} \sim U\left[\frac{-1}{\sqrt{n_{in}}}, \frac{1}{\sqrt{n_{in}}}\right] \tag{2.5}$$

In its turn, in the second paper, it is depicted the famous Xavier initialization, named after its main proponent Xavier Glorot. The advance proposed is to maintain the activation and backpropagation gradient variance, making the uniform distribution also dependent of n_{out}, the size of the subsequent layer. For a hyperbolic tangent function, the random initialization is given by:

$$w_{ij} \sim U\left[\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\right] \tag{2.6}$$

Notwithstanding, assembling both the advantages of Xavier initialization and unsupervised pretraining proposed by Hinton, Saxe *et al.* proposed a random orthogonal initialization for each layer in [23]. In fact, this process is very much related to PCA, as both concepts aim to perform an orthogonal representation of a given dataset; as a note, in [24] a study was done comparing PCA and Xavier initialization in document image analysis: summarily, it has been shown that the former outperforms the latter in convergence speed. Later on, taking advantage of the potential of rectified linear units, He *et al.* presented in [25] a Xavier-inspired initialization method, including the non-linearities of rectifier activation functions.

2.2 Autoencoders applied to the state estimation problem

This section provides a description of the main topics regarding the application of autoencoders in state estimation problems. In order to contextualize the problem, it is firstly presented the general formulation of the state estimation problem and the weighted least squares state estimation method. Then, a brief description of some published works is presented with the purpose of showing the state of the art of the use of autoencoders in state estimation and power system problems.

2.2.1 State Estimation - an overview

In order to successfully overwatch and to increase the controllability of modern-day electrical grids, systems as Supervisory Control and Data Acquisition (SCADA) and Energy Management System (EMS) were designed. This latter system possess a collection of functionalities, including **state estimation**.

First proposed by Schweppe *et al.* in [26–28], state estimation can be defined as the process of determining a valid and highly probable operating point - described by the value of state variables - of the power system based on measurements from the SCADA systems. In other words, state estimation is responsible for filter and eliminate inadequate data and to produce, to a certain point, reliable state estimates for both observable and unobservable parts of the network [29]. Hence, the state estimation problem can be formulated using the subsequent constrained optimization problem:

$$min\ J(x) \tag{2.7}$$

Subject to:

$$c(x) = 0 \tag{2.8}$$

$$g(x) \leq 0 \tag{2.9}$$

Where:

- x is the vector of state variables;
- J(x) is the objective function;
- c(x) and g(x) are the equality-constraint and inequality-constraint vectors, correspondingly;

The objective function, J(x), represents the total error between the the measured and estimated values, parameter which is intended to be minimized. Moreover, having in consideration that a given measurement is equal to its real value plus an error, J(x) can be rewritten using the non-linear measurement model and according to [29]:

$$z_i = h_i(x) + e_i, \quad i = \{1, ..., m\} \tag{2.10}$$

$$J(x) = f(z - h(x)) \tag{2.11}$$

Where:

- x is the vector of state variables of dimension n formed by buses' phase angles and voltage magnitudes;
- z_i is the i^{th} measurement composed by metered line power flows, bus voltages and power injections;
- h_i is the non-linear function that relates the i^{th} measurement to the of state variables, x;
- e_i is the error of i^{th} measurement;
- m is the number of available measurements, with $n < m$;

In the end, the result of this problem will provide an estimation of the real state variables' vector, $\hat{x}$. So, the difference between the measured values z and the estimated measurements $\hat{z} = h(\hat{x})$ will supply an estimate of the errors, frequently called residuals, r:

$$r = z - \hat{z} = z - h(\hat{x}) \tag{2.12}$$

The true value of the measurement error, e, can be defined as the difference between the measured value and the real value corresponding to that measurement. Even so, the real value of the measurement value is unknown, forcing the adoption of the residual in the practical implementation of this algorithm.

Regarding the seminal works by Schweppe *et al.*, it can be declared that the most widely used criterion to solve this problem is the minimization of the weighted sum of the squared residuals, establishing the well known procedures of Weighted Least Squares (WLS) power system state estimation, that is more thoroughly analyzed in the next section.

2.2.2 WLS state estimator

Nomen est omen, the WLS state estimator aims to minimize the square residuals, each one of them associated with a specific weight. Thus, the objective function, $J(x)$, can be expressed as:

$$J(x) = [z - h(x)]^T R^{-1} [z - h(x)] \tag{2.13}$$

Matrix R is the covariance matrix of measurement errors and assuming that measurement errors are independent, R can be portrayed as $R = diag \sigma_1^2, ... \sigma_m^2$ [30]. On the other hand, R is the equivalent of W^{-1}, being W the diagonal matrix whose elements are the measurement weights. This weights may be seen as a set of parameters which specify the degree of trustworthiness in each one of those measurements.

To solve this optimization problem, it is generally used the Newton-Raphson method. At the minimum, the first-order optimality conditions have to be observed [30]:

$$g(x) = \frac{dJ(x)}{dx} = -H^T(x)R^{-1}[z - h(x)] = 0 \tag{2.14}$$

Where H is the Jacobian matrix:

$$H = \begin{bmatrix} \frac{dh_1(x)}{dx_1} & \cdots & \frac{dh_1(x)}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{dh_m(x)}{dx_1} & \cdots & \frac{dh_m(x)}{dx_n} \end{bmatrix}$$

Rewriting $g(x)$ into its Taylor series around the state vector x^k, it is obtained the following result:

$$g(x) = g(x^k) + G(x^k)(x - x^k) + ... = 0 \tag{2.15}$$

Neglecting higher order terms for the sake of simplification, it is possible to determine:

$$G(x^k)\Delta x^{k+1} = g(x^k) = H^T(x^k)R^{-1}[z - h(x^k)] \tag{2.16}$$

$$\Delta x^{k+1} = x^k - x^{k+1} \tag{2.17}$$

Where:

- k is the number of the iteration;
- x^k is the state vector at the k^{th} iteration
- G(x^k) =$H^T.R^{-1}$.H(x^k)
- g(x^k)=$-H^T.R^{-1}$.[z-h(x^k)]

The matrix G is called the gain matrix and it is characterized by being a sparse, positive and symmetric matrix based on the assumption that the system is fully observable [30].

That being so, an iterative solution scheme is achieved. The algorithm of WLS state estimator stops when $\Delta x^k < \varepsilon$, being ε a defined limit for the maximum state variable deviation, usually called convergence criterion. Thus, the steps necessary to perform an iteration of WLS state estimation procedure are presented below [31]:

1. Set k=0;
2. Initialize the state vector x^k, typically a flat start is used;

3. Calculate the measurement function, h(x^k);
4. Build the measurement Jacobian, H(x^k);
5. Calculate the gain matrix, G(x^k);
6. Calculate the right hand side of the normal equation: $H^T(x^k)R^{-1}$[z-h(x^k)];
7. Solve the normal equation in order to Δx^k;
8. Check for convergence using max $|\Delta x^k| \leq \varepsilon$;
9. If it has not converged, update $x^{k+1} = x^k + \Delta x^k$ and go back to step 3. Otherwise, the algorithm stops.

The proposed method gives reliable results for small measurement deviations and if the necessary information is known beforehand. In fact, there are several actions performed besides the state estimation in itself - such as topology processing, observability analysis and bad data processing - forcing a much wider understanding of the concept of state estimator.

2.2.3 Topology Estimation

On the whole, the previous section states that state estimation provides a "snapshot" of the system's state accordingly with its input data. Besides line power flows, power injections or bus voltages, circuit breaker's statuses - which define the system's topology - are key to the accurate execution of the algorithm. Despite its digital nature, topology information is also susceptible to errors in the same way as the analog measurements of current or voltage, possibly leading to actions that can endanger the system's stability. Therefore, topology errors are observed when there is an inconsistency between the actual state of one or more network components and their modelized state in the topology processor as a consequence of erroneous input data [32].

As a result, a considerable amount of methodologies were proposed with the purpose of detecting, identifying and correcting topology errors since the massive deploy of state estimation tools in power systems. In a direct manner, theses methodologies can be dichotomized in two major categories according with their moment of execution with respect of the state estimation calculation: ***a priori*** and ***a posteriori*** methods.

First works related to this subject [32–35] date back to 80's and preconized *a posteriori* approaches to this problem based on the analysis of the indirect influence of topology errors on the value of the residuals' vector. Then, a repetitive analysis through successive computations of the state estimation algorithm was made for all the combination of states for the suspected branches. Although simple, topology error detection techniques with such guiding principles collide with poor convergence - or even non-convergence - of the WLS method in the event of occurrence of topological errors, eventually jeopardizing the topology tests. Apart from that, it is clear that the above-mentioned methodologies require significant computing times.

Against the dominant point of view of the time, Irving and Sterling in [36] presented in 1982 an *a priori* strategy of topology error detection by local validation of data in substations through constrained-linear programming. Later, and in response of the downsides of *a posteriori* methods, Singh and Glavitsch in [37] also published a rule-based a priori algorithm for topology error identification. These two works constitute the first efforts to establish an *a priori* methodology.

Despite the advancements in this area, Monticelli with [38] truly established a new paradigm regarding the state estimation by integrating topology identification in the state estimation and bad data processing procedures. This approach is depicted in full detail [38] as **Generalized State Estimation** (**GSE**). Previously, articles [39,40] presented a new methodology to represent a given circuit breaker as a short-circuit branch - if closed - or as a branch with no power flow - if open. In doing so, an estimation of the status of the circuit breaker or switch is possible through a classical SE method. Accordingly, the GSE algorithm is described in [38] as two-stage process:

- **Stage 1:** A classical estimation procedure is performed with the purpose of analyzing the system and identify suspected zones .
- **Stage 2:** This stage examines the suspected zones determined before and that remained unscrutinized in the preceding stage

Despite being prone to the downsides of *a posteriori* methods, this GSE procedure has become standard and later publications were inspired by this trend. One of them [41], by Clements *et al.*, reported a way of detecting topological errors through normalized Lagrange multipliers, reformulating the second stage earlier proposed by Monticelli. Afterwards, works along this line of thought such as [42,43] were published showing computational burden improvements. Another approach regarding the GSE principle can be seen in [44] where it is employed a WLS-based fuzzy state estimator.

On the other hand, specific topology determination procedures were developed. One example is [45] in which is utilized a mix of the basic framework provided in [38] and combinatorial bad data analysis via decision trees. Conversely, a methodology was proposed by Caro *et al.* in [46] which used mixed-integer quadratic programming to perform topology identification prior to the state estimation itself.

All the works depicted above can be classified as conventional approaches; i.e. directly linked to the use of mathematical equations like the ones presented in 2.2.1 and 2.2.2. Over the years, other techniques were elaborated taking advantage of the potential of artificial neural networks, being labeled as unconventional approaches.

The key principle behind the use of ANN in this context is rather simple to be explained: given a dataset comprising measurements and their respective topology, the ANN is trained with the objective of associating topology scenarios with their proper measurements. Later on, in real situations, the ANN is "feeded" by measurements performed by SCADA systems and has to determine the topology of the system based on his early training. In doing so, this approach gained popularity due to their fast and reliable results an by being immune to convergence problems of conventional methods.

Pioneering works by Alves da Silva in [47–49] paved the way to a more frequent use of this approach by presenting a combined method for topology identification, observability and bad data analysis. Since then, this kind of solution had other developments, depicted in [50]. Alternatively, Kumar *et al.* in [51] performed a comparison between types of architecture of neural networks and conventional methods within the scope of static state estimation and topology determination. Additionally, in [52] it is presented a fuzzy clustering and pattern matching way of performing an integrated procedure of state estimation.

2.2.4 Application of autoencoders to state estimation

Considering the relevance of state estimation for the proper monitoring and control of power systems, a considerable amount of methodologies were proposed throughout the years aiming new perspectives and improvements to state estimation tools.

Although the adoption of ANN's in the scope of power system state estimation dates back to more than two decades, autoencoders until very recently did not penetrate this particular field. More precisely, it was in 2012 that one paper by Miranda *et al.*, [53] presented, as proof of concept, that AE's specificities could be exploited for state estimation, paving the way to other works that adopted the framework originally provided. Primarily, it is reported the development of an AE-based way of reconstructing missing signals from breaker and metering devices in the context of state estimation using genetic algorithms for optimization procedures. Also, a global vs local AE implementation is addressed, in which similar results in terms of accuracy were obtained but with a great difference in computational effort.

In the following year, Krstulovic in [54] published a decentralized, competitive and mosaic-type solution based on competitive autoencoders in order to successfully identify the topology of a given system. One of the most important guidelines of this work is the full acknowledgment of the local impacts of breaker status. This assumption led to the conclusion that a set of AE's, each one assigned for a specific grid location, yielded much better results in terms of accuracy as opposed to what was reported in [53]. Following a author's recommendation for future work, a distinct training procedure using unsupervised learning techniques and ITL criteria was applied in [12].

Aside from the intent of providing solutions related to topology identification, one can point a less compartmentalized application of autoencoders in power system state estimation. As it is widely known, performing a traditional state estimation procedure in a distribution grid is, in some circumstances, a very hard task due to the frequent grid's bad characterization. In face of this raised issue, a paper [55] proposed the use of AE's for determining the system's state in distribution grids without prior knowledge of grid topology and parameters; a relationship between accuracy and multiple, dispersed AE configurations is also established.

2.2.5 Other applications of autoencoders in power systems

Unlike in the special case of power system state estimation applications, AE-based solutions were applied considerably earlier in other fields of power systems.

With roots in a previous work, [56], which stated the ability of AE's to learn the underlying behaviour of systems and to act as a novelty detector, Martinelli *et al.* proposed in 2004 an anomaly detection tool for power substations [57]. This method was intended to read substations' measurements and identify strange pattern in them, generating an alarm in that case.

On the other hand, reflexes of the typical missing sensor data restoration problem for AE's was portrayed in [58] and [59] with distinct fields of operation. While in the first work is depicted the application of autoencoders as a reconstruction block between measurements acquired by SCADA systems and a wide area state predictor, the second paper displays an adoption of AE's to reconstruct missing measurement signals in non-linear power plants.

Exploring AE's for anomaly detection and classification, in 2010, Miranda *et al.* [60] proposed a methodology to diagnose incipient faults in power transformers merging AE's and ITL cost functions. Specifically, a set of autoencoders were trained so that each one of them was associated with a predetermined fault mode. In real situations, given the input vector, autoencoders compete between them in order to determine the cause of the fault. Furthermore, similar utilizations of AE's are reported *et al.* in [61, 62] for detection of transmission line faults and imminent wind turbine blade breakage.

Other practical appliance of this type of ANN's in power systems is depicted in [63] to wind-hydro coordination: taking into account the difficulties of metaheuristics to solve large scale optimization problems, the key principle is to apply a metaheuristic in a reduced space S' controlling the solution in the original, large scale search dominion, S, through AE's. Even if an improvement in the metaheuristic performance is achieved, this kind of method implies, inexorably, a rather significant computational burden which can be, often, unaffordable.

Presenting other possible application, Kelly and Knottenbelt in [64] proposed the use of autoencoders to perform energy disaggregation - estimation of the electricity consumption of each device of an household through frequency analysis. Resembling some traits of missing data restoration, this work presents an adoption of denoising autoencoders as an attempt of obtaining a "clean" power signal of the target appliance, filtering the "noise" produced by other equipments.

At last, one may also highlight the role of AE's as part of forecasting models. Whereas in [65] stacked denoising autoencoders were applied for feature extraction of wind speed data , in [62] this same category of autoencoders were used for short-term electricity prices forecasting.

2.3 Final Remarks

This thesis proposes to add more knowledge to the application of unconventional techniques to state estimation, reinforcing the value of adopting autoencoders to this particular problem.

Reviewing past works in this field, depicted in 2.2.4, it is clear that almost all of them lie in shallow neural network architectures architectures and quadratic cost functions, namely the MSE criterion. Further methodological developments along this line of thought were rather stalled and attached to the proof of concept described in [53] with few exceptions being in [12] in which are presented developments in training procedures and cost functions and in [55] depicting the use of a resilient backpropagation algorithm. In the author's view, the insertion of crucial changes in these latter aspects as well as in the architecture is decisive to achieve better performances in terms of computational times and accuracy of state and topology estimation.

In a brief manner, the advancement proposed in this work is the employment of AE's with more hidden layers in order to successfully extract more valuable relationships and characteristics, taking advantage of deep learning algorithms. Additionally, cost functions inspired in ITL concepts are going to be used with the purpose of preserving the maximum amount of information through the neural network or autoencoder, complementing at the same time the mission of deep architectures and algorithms of discovering new, useful data. A training process very much in the way of the portrayed in [12] is going to be put into practice, assessing its ability out of the traditional, shallow autoencoder configuration. Furthermore, along with increasingly deeper models, greater computational burdens are expected. As a result, the application of GPU computing to this state estimation approach will be reported in this thesis, with the aim of performing benchmark comparisons between GPU and CPU processing.

Chapter 3

State estimation with Deep Learning and ITL Techniques

This chapter will serve as a deeper introduction of entropy and machine learning concepts, highlighting their relations and practical convenience of their combination.

Moreover, it will be provided a short contextualization about classification problems in deep learning. Given the kind of exercise to which this work is related to (identification and proper distinction of the status of a breaker or switching device), a brief analysis on this family of machine learning problems is required. More properly, notions about the definition of a classification task and the depiction of logistic regression as a suitable algorithm to solve this family of problems will be addressed in the following part of the chapter.

Relying on the previously presented notions, detailed formulations of the two methodologies are going to be put forward in order to clarify not only the training procedures but also mechanisms of the employed ANN structures.

3.1 Formulation of the problem

Estimating is directly correlated with the act of performing an approximate judgment about the size, worth, amount of something. Accordingly, power system state estimation corresponds, as the name implies, to the estimation of the system's state, given by its node voltage phasors and transformer's turn ratios.

However, state estimation is highly intertwined with other power system procedures in order to get a reliable model of the given system. Those other procedures comprise, among other tasks, obtaining measurements, observability analysis, bad data processing and, in the end, topology estimation.

The topology of a system is determined by statuses of all breakers of that same system (i.e., any type of switching devices that influence the network connection), described by binary variables since a breaker can only be in an open or closed state (OFF or ON). Thus, after topology determination for all substations throughout the system, a bus/branch model may be built and a power flow or a state estimation can be run. Topology can be seen as a time-dependent or time-independent problem; i.e., it is possible to consider the current topology of a system as an evolution from other topological scenarios due to technical reasons or to consider the actual topology as a separated frame of a film, unattached to the temporal dimension. In this work, the latter paradigm is adopted.

Since the early beginnings of the adoption of state estimation tools that the main methodologies for topology estimation are intimately related with traditional mathematical programming considering the power flow equations defined by Kirchoff's circuit laws. On the other hand, it is clear that topology greatly affects electrical values in a system. Conversely, it is also possible to understand topology estimation as an information extraction process that identifies the status of a given breaker based on the pattern of the measurements, as it is described in Figure 3.1. To perform this, an ANN approach can be used, persistently sweeping a considerable number of topology and operating scenarios, and, ultimately, make the distinction between open and closed breaker based on previous experience.

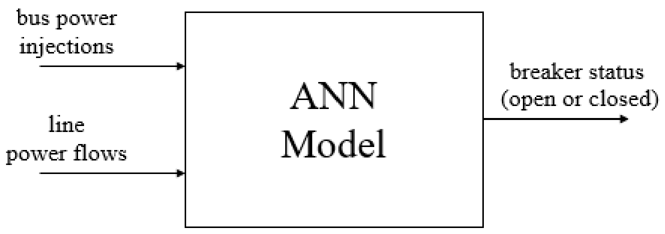

Figure 3.1: ANN-based breaker status processor

All things considered, the main issue is the determination of a system's topology. With this in mind, one must ask: **Can an ANN capture the information hidden in analog measurements and identify a breaker status? If yes, what is the accuracy of that process?**

Therefore, the work developed in this thesis consists in presenting valid, accurate and ANN-based models that take advantage of the relationship between analog measurement manifolds and breaker status from a local, distributed perspective to successfully address those issues.

3.2 Theoretical background

This section will provide a review of the genesis of information theory and its role on the development of ITL. Furthermore, the essential principles of entropy and mutual information along with their relationship between machine learning criteria are exposed.

3.2.1 Brief historical review of information theory

Since the developments of telegraph and telephony in the nineteenth century that communication over lossy and noisy channels is a crucial application which instigated the interest of the most diverse fields of science, specially in mathematics and physics. Despite the notorious advancements in the physical infrastructure of communication systems, there was the need of formulating and applying techniques to the message itself, dealing with the possibility of errors.

Despite earlier works by Nyquist and Hartley in which they, innovatively, shed some light on information-theoretic principles, it is correct to affirm that the article " A Mathematical Theory of Communication" by Claude Shannon, [66], serves as the birth certificate of the discipline of information theory. Rather surprisingly, the work was quickly accepted by the scientific and engineering communities and its fundamentals were promptly put into practice in real problems.

In a simpler way, one can declare that information theory is the field of science which comprises all the processes related to information, since its transmission to its extraction, characterizing, for example, the structure of a message or proposing new compression methods to achieve less distortion over a given communication channel.

However, at this point one might wonder: **how can information theory be useful in machine learning ?** It is widely accepted that the most common criterion in machine learning algorithms relies in the minimization of the mean squared error or, viewing from another perspective, the minimization of the variance of the error distribution. Despite its simplicity, this metric assumes, firstly, that the error distribution is Gaussian and, on the other hand, despises the importance of higher order moments. Given these limitations, new criteria were developed with the essential premise of focusing on the maximization of the information content rather than the steady reduction of the variance of the error distribution, taking advantage of information theory principles.

Concluding and returning to his seminal work [66], Shannon proposed the use of only two descriptors to statistically portray the information character: **entropy** and **mutual information**. These two concepts will be subsequently explored in the next sub-sections.

3.2.2 Shannon's entropy

In an information-theoretic context, entropy is a measure of uncertainty and, at the same time, of quantity of information. Assuming a random variable X, with N pairs $\{x_i, p(x_i)\}$, with a probability density function (pdf) described by $p(X) = \{x_k, p(x_k),\, k = 1,2,...,N\}$, the Shannon's entropy of this random variable is given by:

$$H(X) = -\sum_{k=1}^{N} p(x_k)\,\log_2\, p(x_k) = -E\left[\log_2\, p(X)\right] \tag{3.1}$$

For a continuous random variable, the equivalent expression is followingly stated:

$$H(x) = -\int_{-\infty}^{+\infty} p(x)\log_2 p(x)\,dx \tag{3.2}$$

As it can be seen above, entropy is made from a certain balance in its calculation between the probability and the amount of information: taking a message as example, one can acknowledge that if $p_k = 1$ then its entropy, or information content, is zero and the same rule is applicable if $p_k = 0$. In this latter case, the resulting entropy will be maximum.

This way, Shannon defined the entropy of X as the sum across the set of uncertainty in each component, $\log_2 p(x_k)$, weighted by its probability; if $\log_b$ has $b = 2$, the entropy is measured in bits of information.

3.2.3 Parzen window

The Parzen window technique - also called kernel density estimation - was developed by Emanuel Parzen. Its main goal is concerned with the determination of the pdf of a given random variable from a discrete sample of an M-dimensional space: $y_i \in R^M\ i = 1, 2, ..., N$.

This method consists in applying a kernel function centered on every sample point. Therefore, the estimated pdf, $\hat{f}$, using an arbitrary kernel function is given by the following expression:

$$\hat{f}_Y(z) = \frac{1}{N\sigma} \sum_{i=1}^{N} \kappa \frac{z - y_i}{\sigma} \tag{3.3}$$

where σ is the kernel size or bandwidth parameter. Moreover, a common option for the kernel function type is to choose Gaussian functions. Thus, one can also write the generalized expression for a Gaussian-based kernel estimation of $\hat{f}$:

$$\hat{f}_Y(z) = \frac{1}{N} \sum_{i=1}^{N} G(z - y_i, \sigma^2 I) \tag{3.4}$$

Returning to equation 3.3, the kernel size, σ, is a free parameter which affects heavily the final pdf estimation. Observing the next figure, it is clear that a higher kernel size generates a much smoother form, while lower values of σ provide a much more "spiky" pdf estimate.

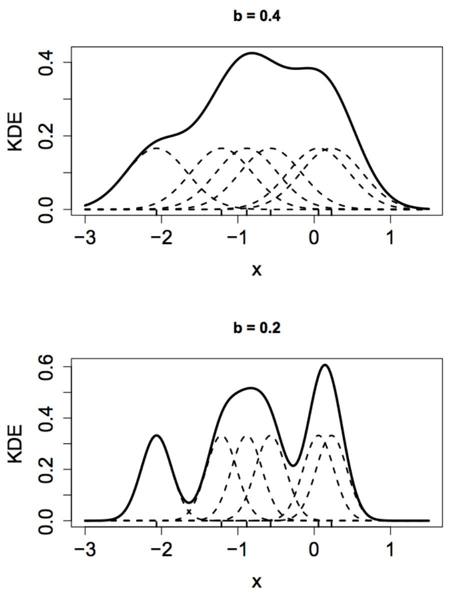

Figure 3.2: Pdf estimation through two kernel sizes represented by b=0.4 and b=0.2.

Due to its importance in kernel density estimation, the generalized determination of the most adequate value of σ was subject of several studies and proposals. To the most part of machine learning applications based on Parzen window, a simple rule of thumb to determine a constant σ value - either Silverman's or Scott's - is enough, although it is possible to use certain heuristics which allow an adaptive kernel size according to the evolution of the cost function [67].

3.2.4 Rényi's entropy

The critical place occupied by Shannon's entropy in information theory as well as its breakthrough character are undisputed. Nevertheless, a more deeper analysis is going to be performed on a distinct approach to the concept of entropy - Rényi's quadratic entropy - due to its applications in machine learning as presented in [13].

Willing to find a more generalized information measure that preserved the additivity of statistically independent systems, the hungarian mathematician Alfréd Rényi proposed a parametric family of entropy functions which generalized Shannon's entropy. Each entropy function is distinguishable by its parameter α and the basic formulation for this family of functions is presented below.

$$H_\alpha = \frac{1}{1-\alpha} \log \sum_{k=1}^{N} p_k^\alpha \; with \, \alpha > 0, \alpha \neq 1 \tag{3.5}$$

Comparing Shannon's and Rényi's entropies, equations 3.1 and 3.5, one can observe that the main difference is the placement of the logarithm in the expression: while in Shannon's entropy the probability mass function weights the $\log_2(p_k)$ term, in Rényi's entropy the logarithm affects the α power of the probability mass function. Also, these contrasts in the mathematical formulation allow to infer that, for optimization algorithms, Rényi's entropy is a much more efficient metric than Shannon's expression. The ultimate reasoning behind this conclusion lies in the fact of Shannon's entropy being composed as a weighted sum of probability logarithms which is computationally harder to solve than Rényi's logarithm of the sum of probabilities.

Given its properties and practical applications, Rényi's quadratic entropy ($\alpha = 2$) possesses a significant and special interest. Applying $\alpha = 2$ to equation 3.5, the quadratic variation of Rényi's entropy is represented by:

$$H_2 = -\log \sum_{k=1}^{N} p_k^2 \tag{3.6}$$

Therefore, giving a geometric interpretation to the formula above, assuming a discrete random variable with a pdf given by $x_i, p(x_i)$ for k=1,2..., n, that same pdf can be seen as a point of a probability space of n dimensions being at the same time over a hyperplane in which the sum of all dimensions - or probabilities - is equal to one. In the particular case of Rényi's quadratic entropy, it corresponds to the negative logarithm of the Euclidean distance of that point to the center.

In machine learning, entropy can be specially useful and appropriate in situations in which data representations are not precisely defined - such as unsupervised learning tasks - because entropy guides the optimization through an infomax principle. Therefore, assuming a continuous random variable, y, and its pdf given by $f_Y(z)$, and combining both Renyi's quadratic entropy and Gaussian kernel pdf estimation, it is possible to construct an entropy estimator as expressed below:

$$H_2(y) = -\log \int_{-\infty}^{+\infty} f_y^2(z)\, dz = -\log V(y) \tag{3.7}$$

$$V(y) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(z - y_i, \sigma^2 I)\, G(z - y_j, \sigma^2 I)\, dz \tag{3.8}$$

V(y) is called the **information potential** and, according with the convolution property of the Gaussian functions, can be written as:

$$V(y) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(y_i - y_j, 2\sigma^2 I) \tag{3.9}$$

Attending to these last equations, one can state that if the information potential, V(y), is minimized, then the entropy, $H_2(y)$, will be maximized, assuring a maximum flow of information through the model. As a result, it is called the **Maximum Entropy Criterion** (MEC).

3.2.5 Mutual information

Through the use of entropy one can describe the amount of information content of a single information source; in other words, entropy measures uncertainty of only one random variable. However, communication systems have inputs and outputs, each uncertain in their own way, so there is a need for a descriptor which has the ability of characterizing two information sources. That same descriptor has been developed by Shannon and it was named mutual information.

Mutual information can be seen as a correlation index between two random variables and measures the amount of information obtained about one random variable, through the other random variable. Formally, considering $X = \{x_k\}_{k=1}^{N}$ and $Y = \{y_k\}_{k=1}^{N}$ two discrete random variables, mutual information between X and Y is given by

$$I(X,Y) = \sum_{i} \sum_{k} p(x_k, y_i) \log_2 \frac{p(x_k|y_i)}{p(x_k)} \tag{3.10}$$

Although mutual information is a distinct descriptor from entropy, the two concepts are intrinsically connected due to their relationship with information content. Accordingly, one can obtain the mutual information between two random variables using the entropy of each one of them. This can be perceived by Venn's diagram below:

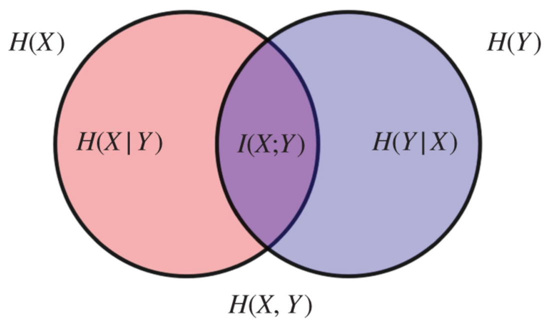

Figure 3.3: Diagram to describe the relationship between entropy and mutual information of two random variables.

The mathematical equation that shows the relationship between mutual information and marginal entropy of X and Y is given by:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{3.11}$$

In accordance with the equation 3.11, if X and Y are independent from each other the the mutual information between these two two random variables will be zero, i.e. each random variable contains no information about the other. Maintaining the same assumption, the joint distribution of both X and Y is given by the product of their marginal distributions. Therefore, this descriptor can be formulated as a divergence between the joint distribution and the product of marginal distributions - if both terms are equal, variables are independent resulting in mutual information equal to zero [12].

As a result, one can use such conclusion to create an information-theoretic estimator - apart from maximum entropy - to act as regulator of an unsupervised learning process in a machine learning algorithm, calculating and maximizing the mutual information between the input and the output pdf of a determined model.

In the field of statistics, divergence is a function that indicates the distance between two pdf and many metrics have been proposed to express such relationship. In this work, it is going to be employed the Cauchy-Schwarz distance to create the desired estimator. Subsequently, it is applied this same definition to a situation where there is two distinct pdf with only one random variable:

$$I_{CS}(f,g) = \log \frac{\int f^2(x)\,dx \;\int g^2(x)\,dx}{(\int f(x)g(x)\,dx)^2} \tag{3.12}$$

Suiting this same formula to the case when there is two random variables, one gets:

$$I_{CS}(X,Y) = \log \frac{\int\int f_{X,Y}^2(x,y)\,dx\,dy \;\int\int f_X^2(x)\,g_Y^2(y)dx\,dy}{(\int\int f_{X,Y}(x,y)f_X(x)f_Y(y)\,dx\,dy\,)^2} \tag{3.13}$$

The above expression can be seen as the Quadratic Mutual Information (QMI) and may be decomposed in three terms: V_J, V_M and V_C. V_J, V_M represent the information potentials of the joint pdf and of the factorized marginal pdf, respectively; on the other hand, V_C corresponds to the generalized cross information potential. That same partition is observed subsequently:

$$\begin{aligned} V_J &= \int\int f_{X,Y}(x,y)\,dx\,dy \\ V_M &= \int\int (f_X(x)f_Y(y))^2 dx\,dy \\ V_C &= \int\int f_{X,Y}(x,y) f_X(x) f_Y(y)\,dx\,dy \end{aligned} \tag{3.14}$$

The estimator for the first parcel, V_J, is given below:

$$\begin{aligned} \hat{V}_J &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{V}_1(i,j)\,\hat{V}_2(i,j) \\ \hat{V}_k(i,j) &= G_{\sqrt{2}\sigma}(x_k(i)-x_k(j)) \end{aligned} \tag{3.15}$$

Regarding V_M, its estimator is showed in the following equation:

$$\begin{aligned} \hat{V}_M &= \hat{V}_1\hat{V}_2 \\ \hat{V}_k(i,j) &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\hat{V}_k(i,j)\,,\, k=1,2 \end{aligned} \tag{3.16}$$

At last, the estimator for the generalized cross information potential, V_C, is represented as:

$$\begin{aligned} \hat{V}_C &= \frac{1}{N}\sum_{i=1}^{N}\hat{V}_1(i)\hat{V}_2(i) \\ \hat{V}_k(i) &= \frac{1}{N}\sum_{j=1}^{N}\hat{V}_k(i,j)\,,\, k=1,2 \end{aligned} \tag{3.17}$$

Concluding, the I_{CS} estimator considering two random variables, X and Y, is expressed as the combination of the previously presented estimators. Maximizing this estimator constitutes the **Maximum Quadratic Mutual Information Criterion** (MQMIC):

$$\hat{I}_{CS}(X,Y) = \log\hat{V}_J - 2\log\hat{V}_C + \log\hat{V}_M \tag{3.18}$$

Intuitively, under some circumstances maximizing the quadratic mutual information between the input and output of the model is equivalent to maximize the entropy at the output of the same model [12]. The cornerstone assumption behind this conclusion is that in both approaches the objective is to retain the most important characteristics about the input data in the model, being possible to apply two different, although intertwined, criteria. Thus, the choice of the implemented criterion relies only on the straightfowardness of programming and on the computational burden required. In chapters 3.3 and 3.4, where both methodologies are going to be scrutinized, it is reported the adoption of Maximum Entropy Criterion (MEC) in both cases.

3.2.6 Machine Learning Classifiers

Due to its moldability, machine learning is a powerful and appalling solution to a broad range of problems, involving the fields of computer vision, pattern recognition and automated learning. Even though the enumeration of such problems could be a slow and time-consuming quest, one can divide such problems by categories based on the type of task that one desires to perform. Hence, possible applications of machine learning can fit in one or more of the following common designations: classification, regression, transcription, machine translation, structured output, anomaly detection, imputation of missing values, denoising, etc.

However, the particular interest regarding the methodologies which will be later presented - the distinction of open or closed state of a breaker device - is classification. To delineate precisely this type of problem, one can quote an adequate definition from [1]:

> "Classification: In this type of task, the computer program is asked to specify which of k categories some input belongs to. To solve this task, the learning algorithm is usually asked to produce a function $f : R^M \ \{1, . . . , \mathrm{k}\}$. When $y = f(x)$, the model assigns an input described by vector x to a category identified by numeric code y."

Putting it more simply, classification consists in assigning a specific label to a case based on the inputs fed to the computer program; possible, everyday examples of such problems are object recognition, handwriting recognition or medical image analysis.

If in some cases there were disadvantages which had to be surpassed, in others there had to be a clear response to some unique and idiosyncratic aspects of the given problem. In order to accomplish those goals, a multitude of classification algorithms were developed. Among them, there are unanimous and long-established groups of procedures such as decision tree-based models, supporting vector machines, kernel estimation classification methods or linear classifiers. Given the adoption of a linear classification algorithm in the second methodology, a more in-depth analysis is going to be performed, in a first place, on this family of methods and then to logistic regression.

3.2.7 Logistic Regression

Linear classifiers perform classification through the evaluation of the algebraic, linear combinations of the input. Regarding the perspective from machine learning, this is a definition which suits to logistic regression: this same model is composed by W, a weight matrix, x, an input vector and b, a bias vector. Classification is done by projecting the input vector to a set of hyperplanes, each one of them corresponding to a class. The distance from the input to a hyperplane reflects the probability of the case denoted by the input vector being a member of that specific class.

Thus, the adopted model of logistic regression can be defined by the following equation. The probability of an input vector, x, belonging to a class i can be expressed by:

$$\begin{aligned} P(Y = i\,|W,b) &= softmax_i(Wx + b) \\ &= \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}} \end{aligned} \tag{3.19}$$

Being:

- $Softmax$ - terminology used to represent the softmax function
- Y - output stochastic variable which indicates the membership of an input vector to a certain class

However, the previous formulation shows the probabilities of a given case of belonging to every class. The prediction of the most likely class to which the input vector belongs is given by the class with maximum probability, $max\,P(Y = i|W,b)$.

In machine learning and more accurately in the neural networks universe, finding the optimal parameters for a model corresponds to a loss function minimization. In case of multi-class logistic regression it is very common to employ the negative log-likelihood as the loss, maximizing, in practical terms, the likelihood of the data set under the characteristics of the given model. Formally, one can express the criterion as:

$$\begin{aligned} \mathcal{L}(\theta = \{W,b\}, \mathcal{D}) &= \sum_{i=0}^{|D|} \log(P(Y = i^{(i)}|x^{(i)}, W, b)) \\ \ell(\theta = \{W,b\}, \mathcal{D}) &= -\mathcal{L}(\theta = \{W,b\}, \mathcal{D}) \end{aligned} \tag{3.20}$$

Where:

- $\mathcal{D}$ - corresponds to the dataset
- θ - symbol which represents the model being optimized
- $\mathcal{L}$ - log-likelihood estimator
- ℓ - loss function to be applied, negative log-likelihood

The main reasons for using this type of classification algorithm lie predominantly in the easy and robust update of the parameters via a stochastic gradient descent method, providing, at the same time, an understandable probabilistic approach.

3.3 Methodology 1 - Deep AE

This section is intended to show explicitly the structures and methods used to construct the first variant of ANN training through the utilization of deep autoencoders.

In order to show the underlying applied techniques, weight initialization, optimizer methods and architectures are issues which will be addressed in this subsequent part of the chapter.

3.3.1 Overview

Prior to the details related to training autoencoders, it is a priority to define the bottom line of this first methodology, mainly in the aspects of range of the topology estimation and the assumed principles to perform such task.

Regarding the character of topology estimation using autoencoders, it is possible to discriminate two possible alternatives: global and local. These two variants are presented and specially focused on chapters 2.2.4 and 2.2.3 of this thesis. Summarily, global topology estimation proceeds to the determination of topology in one stroke, willing to distinguish the status of all breaker devices at the same time. On the other hand, local topology estimation assumes a more decentralized, mosaic-type solution: in this case, adopting the use of two autoencoders for each breaker device - one for open state and the other for closed state -, one can get the topology of the system through the contribution of every pair of AE's.

Each one of this approaches has its own advantages and downsides. For example, the fact of putting into practice a variant of global estimation assures a much more automated and centralized procedure because it is only necessary to design one ANN structure. Nevertheless, acknowledging that information about the breaker status possesses remarkable local properties and that the amount of data and training times can be significantly lower, it was chosen to stick with a local strategy rather than a global one.

Therefore, one can summarize such local strategy as follows:

- An AE is trained to learn a pattern of electric measurements when the breaker is closed; on the other hand, another AE is trained taking into account the electric measurements when the breaker is open.
- To test the generalization capacity of both autoencoders, it is given to them an input with a case that doesn't figure in training or validation sets.
- Whereas one of the autoencoders should recognize such inputs as belonging to the learned manifold, the other will show a larger reconstruction error due to the fact of not being prepared for such example.
- Selecting the AE with the lower error corresponds to the prediction of the open or closed state of breaker

This kind of perspective for local topology estimation has been described as **competitive autoencoders** and is represented on Figure 3.4. The rationale behind this denomination is the constant and adversarial error comparison between the two autoencoders in order to determine the breaker or switching device status.

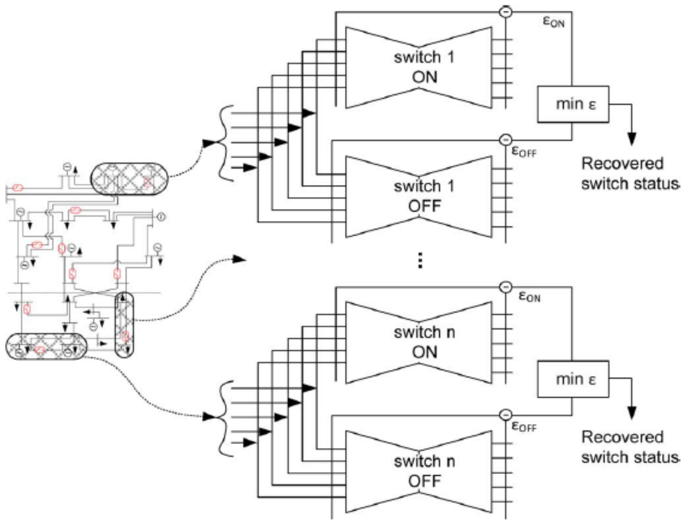

Figure 3.4: Illustration of competitive local autoencoders - from [54]

As perceived in earlier published works, the two dominant aspects regarding the precision of breaker status estimation are the constitution of the training, validation and test sets and the training. Hence, these two fields will be depicted in the following sections.

3.3.2 Dataset and Pre-processing

Despite the previous description of the process to be implemented, it is absolutely necessary the existence of electric measurement data to serve as input to the ANN architectures.

As a test system, it was chosen the IEEE RTS-24 to serve as the example for the further studies with ANN structures presented in this thesis. Additionally, the case study to which both methodologies were applied is the identification of the open or closed status of breaker 2. A more thorough assessment of this two aspects is going to performed ahead in Chapter 4.

In order to obtain the database comprising all the studied cases, the following steps were executed:

1. Design of an annual cumulative load curve based on data from [68]; then, load levels are randomly sampled.
2. Simulation of load variation through the addition of a Gaussian perturbation with $\sigma = 5\%$.
3. Generation of a large set of realistic operating scenarios with an OPF, with breaker status randomly defined.
4. Simulation of noisy measurements by adding a Gaussian perturbation to power flow solutions, with $3\sigma = 1\%$.

The successive OPF experiments allowed to constitute a database of 11736 examples, as it can be seen in table 3.1, which were splitted under a 60%-20%-20% partition scheme for training, validation and test, respectively. Moreover, due to the need of training two different autoencoders to compose the competitive scheme, it was created a training and a validation set for each autoencoder; i.e., for each possible state of the breaker.

Table 3.1: Dataset division regarding the first methodology.

		Number of examples
Open State	Training Set	3496
	Validation Set	1165
Closed State	Training Set	3496
	Validation Set	1165
Test Set		2414
Total Number of Examples		11736

Before inputting these sets of raw data into the AE's, it was performed a pre-processing step which consisted in applying a normalization function that transformed every example - row - in an unit norm vector. Despite the wide utilization of this approach in text classification and clustering problems, the use of this normalization procedure in this case is backed by successive experiments with other pre-processing functions such as Gaussian standardization - transform every column by removing the mean and scaling to unit variance - and min-max scaler method for intervals [-1,1] and [0,1]. In the end, unit norm row normalization presented better results than the other functions in terms of estimation accuracy.

3.3.3 Training and model characteristics

As mentioned earlier, the case study is the estimation of the open or closed status of breaker 2. So, taking into account the local character of this methodology, the input data of the autoencoder is only related to the area of influence of that same breaker. In a first approach, it was used an input vector of 22 inputs composed by voltage magnitude and phase of buses 3 and 9, active and reactive power injection in those buses and active and reactive power flows in every line connected to bus 3 and 9 - lines 1-3, 3-9, 3-24, 4-9, 8-9, 9-11 and 9-12. However, the final input vector comprises only the active and reactive power injection in buses 3 and 9 as well as active and reactive power flows of the branches connected to the same buses; this way, the input vector was shortened in 4 measurements, passing from a length of 22 to 18 elements. Although this removal of analog measurements can be seen, in a first glimpse, as an harmful information disposal, this measure has actually positive impact on prediction accuracy as showed in annex A.

As a development platform to construct deep learning and neural network models, the choice has been to adopt Python programming language, taking advantage of libraries such as Theano and Lasagne. These libraries are well-known and widely used in the field, having plenty of documentation for applications similar to the ones presented in this thesis.

The final model is composed by two autoencoders, each one with an input layer of 18 neurons and having 5 hidden layers with sizes of 16, 14, 12, 14 and 16 neurons, respectively. Due to the fact of being a symmetric autoencoder, the output layer has also 18 neurons. Each neuron has a sigmoid activation function.

The success of any ANN structure is dependant of the weights' starting point. Hence, the strategy adopted consists in a random orthogonal weight initialization using a Gaussian distribution between 0 and 1, described in [23]. This option has provided better accuracy results when compared to the other approach, Xavier initialization. Also, other major motive behind this choice is that this approach helps more substantially model regularization; i.e. prevention of overfitting by assigning low values to weights and biases.

The chosen optimization algorithm was ADADELTA - described in full detail in [69] - whose parameters are configured according the existent technical literature. The major reasons for opting for this specific algorithm are the faster convergence and higher capacity of escaping local minima when compared to traditional minibatch SGD along with its ability of adapting the learning rate through the optimization process. Additionally, another important aspect regarding ADADELTA is the absence of the obligation of defining a default learning rate which is a rather common problem in the field of iterative optimization processes.

The employed training procedure and cost functions can be summarily depicted in the following points:

1. Encoder training in a layerwise fashion using a MEC cost function
2. Decoder generalized training using the MSE criterion

Considering the training under the MEC criterion, one has to define the procedure related to the kernel size σ, necessary to the application of the Parzen window to perform the estimation of entropy. Firstly, it was attempted an adaptive strategy in which the kernel size progressed from a larger value to a shorter one in order to further optimize the process, avoiding local minima. However, technical problems related to the programming language and libraries have blocked the adoption of such technique. Therefore, after several experiments it was found that applying constant $\sigma = 0.4$ provided the best results on what concerns prediction accuracy.

The training method used minibatches constituted by 50 examples. The stopping criterion consists in the use of early stopping to prevent overfitting phenomena by monitoring the model's performance on a validation set. Concretely, if the model's performance stops improving or starts to deteriorate the algorithms ceases its optimization process if the additional requirement of the minimum number of examples - patience - is met. In the final version of the model, the patience parameter is defined as 20000.

The evolution which allowed the determination of the ideal model parameters along with the specification of the resulting confusion matrices is explained in full detail in annex A.

3.4 Methodology 2 - Deep MLP

This section has the specific purpose of providing deeper information about the more technical aspects of the second methodology - Deep MLP.

In the same way of the previous section, topics related to weight initialization, architectures or training procedures will be thoroughly examined.

3.4.1 Overview

Subsequently to the idealization and application of the first methodology, two legitimate questions were asked about the possibilities of reducing the execution time of training the proposed autoencoders and maintaining some traits of the reported information-theoretic approach.

Attending to the nature of the task, the perspective of adopting a single MLP combined with a logistic regression layer is very attractive: in a first place, it serves the main purpose of breaker status identification as a classification problem; the other major point is related to the reduction of ANN structures, passing from two AE's to one single MLP, enabling a relevant reduction in computational time.

To conclude, it was also applied a local approach to topology estimation in order to enable fair contrasts between both methodologies. Again, the breaker 2 serves as the visible example of the method.

3.4.2 Dataset and Pre-processing

In order to perform *ceteris paribus* comparisons between both processes, it was opted to use the same data in both Deep AE and Deep MLP methodologies. Even so, given their fundamental differences related to the number of ANN structures used in methodology 1 and 2, some minor adjustments in dataset division had to be executed. While in methodology 1 one had to divide the example database in open and closed state for AE training and validation, in methodology 2 that step is not necessary due to the attributes of the logistic regression classifier.

As a result, it is possible to immediately split the database under the same 60%-20%-20% criterion, as applied before, as long as the data is labeled to allow the algorithm to learn the patterns from open and closed state of the breaker.

Table 3.2: Dataset division regarding the second methodology.

	Number of Examples
Training Set	6992
Validation Set	2330
Test Set	2414
Total Number of Examples	11736

As a pre-processing step, it was performed the application of a normalization of every row to unit norm, in the same way of methodology 1.

3.4.3 Training and model characteristics

Given the fact of both methodologies being local approaches, in methodology 2 it is adopted the use of analog measurements from the surrounding zone of buses 3 and 9. Therefore, in the final version of this methodology it is used the active and reactive power injection in those buses along with the active and reactive power flows in the branches connected to buses 3 and 9. Hence, the resulting input vector is composed by 18 elements.

Besides the input vector, this methodology has other common points with Deep AE, namely on what concerns programming language, the optimization algorithm, kernel size and training mode. Therefore, in a quick description, the used backpropagation algorithm was ADADELTA, the adopted kernel size was a constant σ of 0.4 and the model was trained using a SGD approach with minibatches composed with 50 examples and a minimum umber of analyzed examples - patience parameter - of 10000.

The ultimate version of Deep MLP model has an input layer constituted by 18 neurons, being followed by 4 hidden layers with 14, 10, 6 and 4 neurons, respectively. In the end, the logistic regression layer has a total of 2 neurons to perform the classification task of distinguishing the open and closed states of the breaker.

The training procedure used for Deep MLP resembles a very well-known approach which consists in combination of layerwise pretraining followed by a generalized finetune of all model parameters. Notwithstanding, a minor tweak was introduced making use of ITL notions. Hence, the training strategy can be summarized in the subsequent points:

1. Layerwise pretraining using MEC.

2. Finetune applying the minimization of the negative log-likelihood.

In this second methodology it was a used a combination of random orthogonal and Xavier weight initialization techniques. Initially, it was tried random orthogonal initialization in all layers. However, putting a Xavier initialization in the logistic regression layer has proved itself useful in terms of accuracy.

Below lies a table which offers a resumed perspective about the two methodologies, putting emphasis on the main aspects which identify each approach.

Table 3.3: Comparative table of methodologies 1 and 2.

	Methodology 1	Methodology 2
Architecture	Deep Autoencoder	Deep MLP
Number of ANN Structures	2	1
Training Procedure	Step 1: Layerwise encoder training using a MEC criterion Step 2: Decoder training using a MSE criterion	Step 1: Layerwise pretraining using a MEC criterion Step 2:Finetune using the minimization of negative log-likelihood
Input Layer Size	18	18
Number of Hidden Layers	5	4
Hidden Layer Sizes	16, 14, 12, 14, 16	14, 10, 6, 4
Output Layer Size	18	2
Weight Initialization for Sigmoid Layers	Random Orthogonal Initialization	Random Orthogonal Initialization
Weight Initialization for Logistic Regression Layer		Xavier Initialization
Optimization Algorithm	ADADELTA	ADADELTA
Learning Rate	1	1
Kernel Size	0.4	0.4
Batch Size	50	50
Stopping Criterion	Early Stopping	Early Stopping

Chapter 4

Case Study Results

This chapter will present the results of both methodologies, Deep AE and Deep MLP, to estimate the status of breaker 2 of the modified version of IEEE RTS-24 test system. With this in mind, a short review on the defined conditions of the case study will be, firstly, accomplished.

Besides the analysis of prediction accuracy, an assessment of the running times to train and test both models is going to be performed in order to find contrasts between both approaches. Also, comparisons between the proposed approaches and similar previously presented methodologies of local topology estimation will be done.

Both models were developed in Python programming language, taking advantage of specific deep learning libraries such as Theano and Lasagne. The presented results for estimation of the breaker status of breaker 2 were obtained in a computer with an Intel Xeon E5-1260 v3 at 3.50 GHz CPU, with a 16.00 GB RAM and a Nvidia GeForce TITAN X GPU.

4.1 Description of the case study

As mentioned earlier, in order to evaluate the adequacy of the prediction of both methodologies it was chosen the IEEE RTS-24 [68] power system. The original cause that led to the development of such power system examples was the need of having a standardized platform to test, develop and compare solutions for reliability studies. The IEEE RTS-24 has the advantage of having a wide variety of aspects defined *a priori*. In the addressed problem, OPF simulations were performed in order to build the database which supported the topology studies. Therefore, the following points resume the types of data from [68] which were used in this case study:

- Generation data
- Load model
- Transmission network data

The load model associated to IEEE RTS-24 represents the hourly, daily and weekly load peaks in percentage of the annual peak load value, 2850 MW. A hourly basis will be adopted in both methodologies.

However, some minor adjustments had to be done in order to include the topology estimation, one of them being the addition of symbolic breaker devices - implemented in a low-level as the line status - to incorporate multiple topology and operating scenarios. Furthermore, it was envisioned a set of measurement devices, providing useful information about the magnitude and phase of bus voltages, active and reactive line flows and bus power injections.

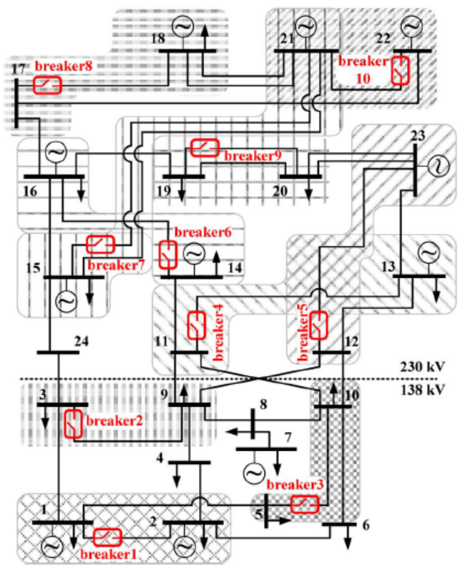

Figure 4.1: IEEE RTS-24 Test Case.

In both training variants, a common principle of local topology estimation is followed. This is done by associating a single ANN or a set of ANN structures to every single breaker or switching device. This way, it is possible to integrate every single contribution to build larger, flexible models.

The total topology of the system is given by the determination of the breaker status of all the 10 breakers depicted in Figure 4.1. In order to execute such task, a total of 20 autoencoders and 10 MLP would have to be trained and tested for methodology 1 and 2, respectively. Such task would be very time-consuming and, in a certain way, dull: to assess the main issues regarding topology estimation based on a ANN approach it is only needed to show the example of one breaker because the process is, substantially, equivalent for each breaker. **Therefore, in the following sections, running times and prediction accuracies will be exposed taking into account the identification of the status of breaker 2, placed on line 3-9.**

4.2 Performance Evaluation of Deep AE

In this experiment, two autoencoders were trained separately, one for open and the other for the closed data cluster, and put in a competitive scheme. The training of this methodology involves a layerwise, unsupervised encoder training based on a maximum entropy criterion being followed by a generalized decoder training minimizing MSE; more specific details about architecture and initialization are fully portrayed in 3.3. The following table summarizes the obtained test results of methodology 1.

Table 4.1: Precision in breaker 2 status diagnosis for methodology 1 - Deep AE.

	Deep AE
True Closed States	1170
False Closed States	5
True Open States	1161
False Closed States	78
Total Precision (%)	96.56

The total elapsed time needed to train both autoencoders and to run the test script for breaker 2 status identification is 85.638502 seconds using only CPU computing and 82.677140 seconds when using the GPU, reflecting a slight acceleration through the use of the latter computational tool. Still related to the computational burden, it would be possible to further reduce the total CPU for the execution of the referred tasks using a special Python library called **multiprocessing**. This mechanism allows spawning multiple processes, fully leveraging multiple processors on a given machine according to user's needs. Concretely, this package would allow the parallel training of the two autoencoders, drastically reducing the total time. However, this tool does not support the possibility of using the GPU; as a result, by not enabling fair comparisons between CPU and GPU computation, this library was not adopted as an official part of methodology 1.

As it can be seen, the application of this approach led to a total of 83 misdiagnosed cases, 5 false closed states and 78 false open states, leading to a total successful identification rate of 96.56%. In a first impression, it is possible to affirm that the technique showed a reasonable accuracy degree, even exceeding a 95% rate of well diagnosed cases.

Before advancing to direct comparisons, it is necessary to highlight fundamental differences between methodology 1 and those works, which are enumerated below:

- Distinct preprocessing procedures: while in this work is preconized an unit-norm row normalization, in the previously mentioned works a minmax scaling between [-1,1] is performed.
- Distinct training strategies and cost functions
- Distinct test set length: while in this work the test set is composed by 2414 cases, in the previously mentioned works the test set is made up from 10000 scenarios.

So, giving the main point for breaker 2 status identification, the table below condensates the percentage of successful diagnosis of other previously approaches:

Despite this evidence, when comparing the results provided in similar studies (depicted in [54] and [12]), it is clear that the determined results of this methodology are not good enough, falling short of those previous approaches. As a result, it can not be stated that this implementation portrays any kind of progress in local topology estimation.

Table 4.2: Highlight of accuracies of past published local topology estimation procedures.

Numerical Reference	Title of Article	Year of Publishing	Total Precision (%)
[52]	Towards an Auto-Associative Topology Estimator	2013	100.00
[12]	Breaker status uncovered by autoencoders under unsupervised maximum mutual information training	2014	99.86

Notwithstanding, to better analyze the results, a data collection was made in order to determine which cases of the test set presented erroneous breaker status identification. That same information is presented below in Table 4.3.

Table 4.3: Specification of failed status diagnosis for methodology 1.

False Closed States	15, 166, 1902, 1998, 2093
False Open States	08, 50, 62, 77, 109, 116, 125, 168, 173, 208, 239, 284, 294, 303, 351, 353, 387, 392, 493, 519, 535, 565, 567, 606, 616, 646, 671, 690, 765, 780, 787, 818, 877, 900, 954, 1000, 1039, 1044, 1092, 1115, 1304, 1355, 1384, 1473, 1488, 1505, 1531, 1532, 1576, 1586, 1627, 1659, 1686, 1691, 1745, 1753, 1755, 1814, 1852, 1873, 1897, 1919, 1982, 2045, 2100, 2119, 2123, 2129, 2161, 2181, 2191, 2197, 2208, 2270, 2291, 2306, 2330, 2397

Whereas the error cause in false closed states was not determined, a possible cause in false open cases could be low branch currents in line 3-9. This way, the algorithm can be confused by such value and give an erroneous prediction - indicating an open breaker when it is, actually, closed. To explicitly show such situations, diagrams with active and reactive power flows in line 3-9 were constructed. The distribution of the 78 false open states in the test set is presented in Figures 4.2 and 4.3.

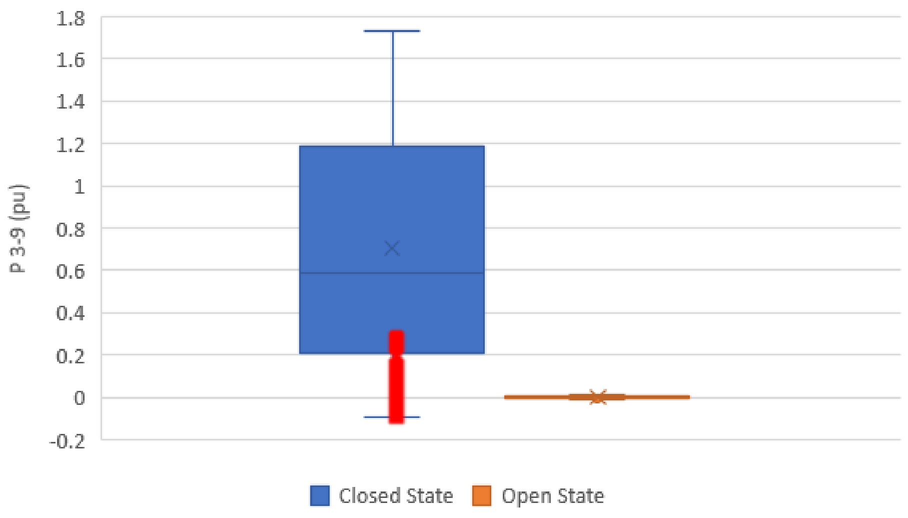

Figure 4.2: Active power flows in line 3-9. In each graph: distribution with closed (left) and open breaker (right). The red dots correspond to the false open states.

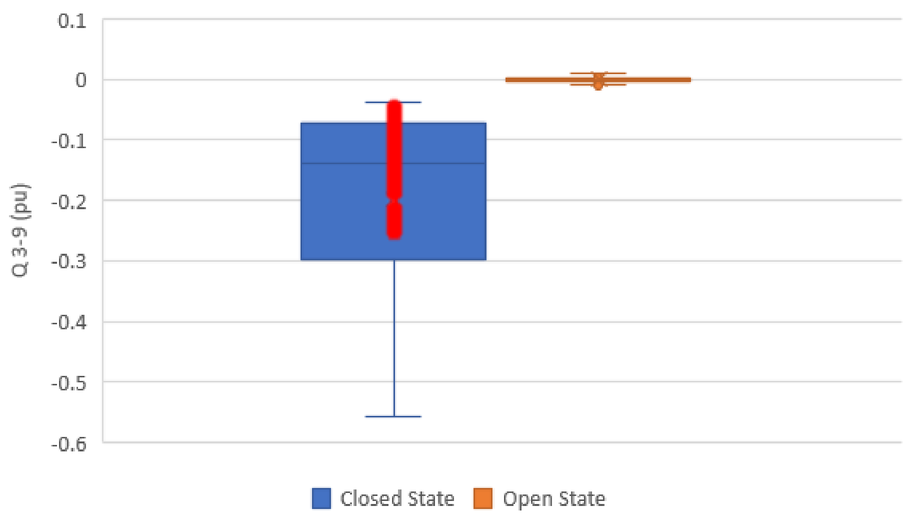

Figure 4.3: Reactive power flows in line 3-9. In each graph: distribution with closed (left) and open breaker (right). The red dots correspond to the false open states.

Observing both box diagrams and specially in the points represented by the red dots, one can notice that all the 78 false open status lie very much below the normal power flows in closed states.

4.3 Performance Evaluation of Deep MLP

Performing a quick review, in methodology 2, a single deep MLP was under a training optimization procedure. This deep MLP was composed by four hidden layers and a final output layer composed by two neurons, performing a logistic regression classification algorithm. The training process is constituted in a first place by a layerwise, unsupervised pretraining guided by a maximum entropy criterion; then, a finetune was performed minimizing the negative log-likelihood at the output. Further specifications about this methodology are included in Chapter 3.4. As in the previous section, the subsequent table outlines the results of methodology 2.

Table 4.4: Precision in breaker 2 status diagnosis for methodology 2 - Deep MLP.

	Deep MLP
True Closed States	1247
False Closed States	0
True Open States	1166
False Closed States	1
Total Precision (%)	99.96

The total elapsed times to train and test this deep MLP are 36.483892 seconds and 21.21959 seconds for CPU and GPU computing, respectively. These temporal measurements reflect a great acceleration: the running time was almost cut in half through the use of the GPU.

Furthermore, it can be observed that there is just one misdiagnosed case in 2414, resulting in a total precision of 99.96%. The example that had a wrong breaker status identification corresponds to the 2045th scenario of the test set. This case was equally mistakenly classified in methodology 1 as being open when, in reality, this scenario is characterized by a closed breaker. Carrying out a manual verification of the active and reactive power flows in line 3-9 in this particular example, one verifies that those values are -0.063302 and -0.048085 p.u., correspondingly. In face of such reduced branch power flows, the erroneous classification is naturally plausible.

Aside from the differences between the processes and tools used in this thesis and past works earlier disclosed in 4.2, there is a fundamental change in neural network architecture. Whereas in methodology 1 and past articles it was implemented a competitive autoencoder scheme, in methodology 2 it was adopted a single deep MLP for each breaker.

Recalling the data exposed in Table 4.2, one can promptly notice that Deep MLP has higher successful identification ratio than in [12], although not presenting a 100% accuracy as exposed in [54]. However, in this latter article it is only adopted a competitive local autoencoder scheme along with a MSE-based training process, not implementing any kind of ITL principle in autoencoder parameter optimization. As a result, one can infer that methodology 2, Deep MLP, has shown some progress towards an ITL-supported, neural network approach for topology estimation.

4.4 Models comparison

In the past sections, direct comparisons between the results of each methodology and past published approaches were accomplished and conclusions were drawn regarding the progress in terms of correct breaker status diagnosis. Therefore, it is time to emphasize the relative performance of Deep AE and Deep MLP. The subsequent tables show the obtained results in terms of time and accuracy.

Table 4.5: Confusion matrices for Deep AE and Deep MLP.

	Deep AE	Deep MLP
True Closed States	1170	1230
False Closed States	5	0
True Open States	1161	1115
False Open States	78	1
Total Precision (%)	96.56	99.96

Table 4.6: Running times for train and test of both models.

	CPU (s)	CPU+GPU (s)	Acceleration GPU
Deep AE	85.638502	82.67714	1.04
Deep MLP	36.483892	21.214959	1.72

Although through past sections it was already clear that the second methodology has a higher successful breaker status identification rate and a better computational times than the first, the tables above give a much more direct perspective of the superiority of Deep MLP over Deep AE.

Besides the error rate of approximately 0.04% of the second methodology, it is necessary to pay close attention to its temporal measurement. While in Deep AE the times for CPU and GPU are practically equal, the time needed to train and test the second methodology decreases to nearly a half when using GPU computing.

Such differences in GPU acceleration are related to training procedures of both methodologies. In Deep AE, one autoencoder is trained after the other and the most prolonged task is the layerwise encoder training following a maximum entropy criterion. Given these two aspects, it is obvious that this solution offers almost no possibility of process parallelization due to its, fundamentally, sequential character. On top of that, the part of the second methodology training which was more time-consuming was the generalized finetuning based on the minimization of the negative log-likelihood; this happens to be the part which benefits the most of GPU because of its capability of parallelizing calculations from different layers, resulting in emphatic computational time reductions.

Chapter 5

Conclusions and future work

5.1 Conclusions

Topology estimation is subject of study since the early days of modern power systems. Furthermore, given the possible and particularly harmful consequences of topology errors for power system stability, topology error detection was depicted in several articles, very much analyzed by the technical and scientific communities.

Succintly, one can affirm that the two main objectives have been achieved. On one hand, it has been proved that the combination of ITL cost functions along with the optimization of hyperparameters of both neural network models lead to breaker status predictions with great level of accuracy. On the other hand, noticeable time reductions in neural network training have been achieved through GPU computing, assessing the potential of this computational tool in the scope of a concrete power system application.

In spite of exposing, undoubtedly, remarkable results, it is imperative to examine the conditions and the nature of the success of both methodologies, specially Deep MLP. Furthermore, only the true, factual application of this type of solutions in real power systems could provide the necessary conditions to proper assess their usefulness.

In order to clearly outline the essential contributions of this work, it is presented below a list of the most important achievements and conclusions:

- Two methodologies - Deep AE and Deep MLP - with new and enhanced characteristics based on a local, decentralized and flexible approach to topology estimation were proposed and tested in very realistic conditions.

- Both methodologies have exposed very good results with success rates in breaker status identification above 95%. However, the more traditional methodology - Deep MLP - performs better than the presumably more innovative one, Deep AE. This better performance is markedly distinguishable in two aspects: total status prediction precision and the overall time for training and testing the ANN structures. At last, it must be noted that the model optimization was coherent in both CPU and GPU.

- A more pronounced time reduction was observed in Deep MLP than in Deep AE methodology when using GPU computation. This has to do, mostly, with training processes and adopted criteria of both methodologies. As a result, it is correct to affirm that the helpful, inherent infomax principle comprised in ITL cost functions comes at a cost: the lack of true capability of parallelization when applying GPU computing due to their layerwise nature.

- The use of classification algorithms serves better the purpose of local topology estimation as breaker status diagnosis when compared to the first methodology; in other words, the distinction between two classes of events is a more appropriate way of dealing with the problem than a error comparison between two autoencoders.

- Unlike what has been affirmed in past works [12], it has been concluded that a more classic approach, such as Deep MLP, does not need any additional external parameter definition regarding a splitting threshold between both open and closed cases.

- In this work, it was felt only one major and remarkable difficulty on breaker status prediction: diagnosing closed states with low branch current. Therefore, significant developments in pattern recognition in these situations must be pushed forward in order to progressively achieve 100% accuracy rates.

5.2 Future work

Due to an inexorable time limitation, it is not always possible to address every single captivating aspect when developing a master thesis. Thus, it is presented subsequently a short list of further developments that can be added to this work:

- Even achieving lower training times with the GPU, the results presented in Chapter 4 still reflect a very reduced acceleration. Hence, in order to further assess the usefulness of GPU in topology estimation, it would be interesting to build larger models, using more hidden layers with a higher number of neurons and more extended input vectors, to highlight the conditions of enhanced performance with GPU computing.

- In the explored case study, the estimation of status of Breaker 2 in line 3-9, the values of active and reactive power flows of line 3-9 are included in input vector. These values constitute a very powerful information which in some cases may undermine the consistency of the learning algorithm. As a result, it would be very interesting to evaluate the methodologies' performance without those electric variables. Furthermore, one can extent this criterion and observe the accuracy of the proposed methods in conditions of low redundancy; i.e., with missing measurements.

- In Chapter **??** it was declared that the dataset was composed of successive OPF experiments, generating a large database of operating scenarios. However, OPF algorithms obey to specific rules and will always privilege higher power productions in generators with cheaper costs. This situation does not correspond to modern, free energy market operating scenarios and introduces a bias in the dataset construction. So, by the construction of a database with randomly defined loads and generation, one could perform an experiment of testing the presented topology estimation methods in market environment, studying, at the same time, the reliability level of such solutions.

- An interesting, useful line of investigation could be the exploration of a local approach with new ITL cost functions to achieve a total unsupervised learning paradigm for a topology estimation application. This way, the problem would resemble a clustering problem in which one would try to aggregate all the open cases and closed cases in different regions of space.
- In this work, the power system behaviour is assumed as being in steady state. This way, we are performing an analysis about the system's topology based on a "snapshot" of the system. But what if this analysis was supported by the "film" of system's behaviour? Therefore, the development of a topology estimator which takes into account past topological sequences and, in general, considers the temporal aspects of topology would lead to a significant progress in this area.
- Alternative programming languages, libraries and classification algorithms could provide new possibilities of building deep learning models with enhanced performance and process parallelization.

Appendix A

Tables of Methodology 1

This annex presents a sequential evolution of the hyperparameters of the first methodology, Deep AE. The last configuration, configuration 10, corresponds to the final, applied version of autoencoders to perform topology estimation on breaker 2.

Table A.1: Hyperparameters for the first five configurations of Deep AE.

	Configuration 1	Configuration 2	Configuration 3	Configuration 4	Configuration 5
Input Layer Size	22	20	18	18	18
Number of Hidden Layers	3	3	3	3	5
Hidden Layer Sizes	18, 16, 18	18, 16, 18	16, 14, 16	16, 14, 16	16, 14, 12, 14, 16
Decoder Training	Joint	Joint	Joint	Joint	Joint
Batch size	100	100	100	100	100
Patience	10000	10000	10000	10000	15000
Kernel size ()	0.4	0.4	0.4	0.6	0.4
Total Precision (%)	78.87	90.60	92.83	87.95	96.02

Table A.2: Hyperparameters for configurations 6-10 of Deep AE.

	Configuration 6	Configuration 7	Configuration 8	Configuration 9	Configuration 10
Input Layer Size	18	18	18	18	18
Number of Hidden Layers	5	5	5	5	5
Hidden Layer Sizes	16, 14, 12, 14, 16	16, 14, 12, 14, 16	16, 14, 12, 14, 16	16, 14, 12, 14, 16	16, 14, 12, 14, 16
Decoder Training	Joint	Joint	Joint	Only Decoder	Only Decoder
Batch size	100	50	25	50	50
Patience	20000	15000	15000	15000	20000
Kernel size ()	0.4	0.4	0.4	0.4	0.4
Total Precision (%)	95.22	96.35	96.15	95.98	96.56

Additionally, there are the tables that support the total precision values above. The number of true and false identifications of both open and closed breaker states are stated below.

Table A.3: Representation of the confusion matrices of the first five configurations of Deep AE.

	Configuration 1	Configuration 2	Configuration 3	Configuration 4	Configuration 5
True Closed States	1003	1120	1075	957	1188
False Closed States	265	99	0	0	36
True Open States	901	1067	1166	1166	1130
False Open States	245	128	173	291	60
Total Precision (%)	78.87323944	90.5965203	92.83347142	87.94531897	96.02319801
Closed State Precision (%)	80.36858974	89.74358974	86.13782051	76.68269231	95.19230769
Open State Precision (%)	77.27272727	91.50943396	100	100	96.91252144

Table A.4: Representation of the confusion matrices of configurations 6-10 of Deep AE.

	Configuration 6	Configuration 7	Configuration 8	Configuration 9	Configuration 10
True Closed States	1206	1187	1190	1166	1170
False Closed States	72	27	35	15	6
True Open States	1064	1139	1131	1151	1159
False Open States	42	61	58	82	79
Total Precision (%)	95.21812081	96.35459818	96.14747307	95.98177299	96.56172328
Closed State Precision (%)	96.63461538	95.11217949	95.3525641	93.42948718	93.75
Open State Precision (%)	93.66197183	97.68439108	96.99828473	98.7135506	99.57118353

Appendix B

IEEE RTS-24

The hourly model which supported the generation of the training, validation and test database is calculated according the following tables. The annual peak load is 2850 MW.

Table B.1: Weekly Peak Load in percent of annual peak.

Week	Peak Load	Week	Peak Load
1	86.2	27	75.5
2	90	28	81.6
3	87.8	29	80.1
4	83.4	30	88
5	88	31	72.2
6	84.1	32	77.6
7	83.2	33	80
8	80.6	34	72.9
9	74	35	72.6
10	73.7	36	70.5
11	71.5	37	78
12	72.7	38	69.5
13	70.4	39	72.4
14	75	40	72.4
15	72.1	41	74.3
16	80	42	74.4
17	75.4	43	80
18	83.7	44	88.1
19	87	45	88.5
20	88	46	90.9
21	85.6	47	94
22	81.1	48	89
23	90	49	94.2
24	88.7	50	97
25	89.6	51	100
26	86.1	52	95.2

Table B.2: Daily peak load in percent of weekly peak.

Day	Peak Load
Monday	93
Tuesday	100
Wednesday	98
Thursday	96
Friday	94
Saturday	77
Sunday	75

Table B.3: Hourly peak load in percent of daily peak.

Hour	Winter Weeks 1-8 & 44-52		Summer Weeks 18-30		Spring/Fall Weeks 9-17 & 31-43	
	Weekday	Weekend	Weekday	Weekend	Weekday	Weekend
12-1 am	67	78	64	74	63	75
1-2 am	63	72	60	70	62	73
2-3 am	60	68	58	66	60	69
3-4 am	59	66	56	65	58	66
4-5 am	59	64	56	64	59	65
5-6 am	60	65	58	62	65	65
6-7 am	74	66	64	62	72	68
7-8 am	86	70	76	66	85	74
8-9 am	95	80	87	81	95	83
9-10 am	96	88	95	86	99	89
10-11 am	96	90	99	91	100	92
11-Noon	95	91	100	93	99	94
Noon-1 pm	95	90	99	93	93	91
1-2 pm	95	88	100	92	92	90
2-3 pm	93	87	100	91	90	90
3-4 pm	94	87	97	91	88	86
4-5 pm	99	91	96	92	90	85
5-6 pm	100	100	96	94	92	88
6-7 pm	100	99	93	95	96	92
7-8 pm	96	97	92	95	98	100
8-9 pm	91	94	92	100	96	97
9-10 pm	83	92	93	93	90	95
10-11 pm	73	87	87	88	80	90
11-12 pm	63	81	72	80	70	85

Appendix C

Article for submission

Regarding the innovations in training procedures of autoencoders and their concrete results in precision of estimating the case study's topology, it was decided to write an article.

In short, this article constitutes a summarized version of this thesis: in a first place, a contextualization about the cornerstone principles about state estimation and ITL criteria is made being followed by an explanation about the methodologies used and their outcomes concerning the topology estimation of the modified IEEE RTS-24 test system.

Training Autoencoders for State Estimation in Smart Grids

Rui Oliveira, Vladimiro Miranda, *Fellow, IEEE,* and Ricardo Bessa

***Abstract*—This article aims to present a novel perspective of the application of neural networks, deep learning and Information Theoretic Learning (ITL) to the problem of topology determination and, ultimately, to state estimation. Blending these concepts, the main objective is to predict the topology of a given power system based on pattern recognition of analog measurements. More specifically, two alternative architectures and training procedures - Deep AE and Deep MLP - will be compared in terms of prediction accuracy. Additionally, benchmark comparisons between CPU and GPU execution times will be scrutinized.**

***Keywords*—State estimation, power system topology, Information Theoretic Learning (ITL), artificial neural networks (ANN), autoencoder (AE), multilayer perceptron (MLP), deep learning**

I. Introduction

State estimation corresponds to the prediction of the values of electrical variables of a power system. Traditionally supported by mathematical equations, this process is intended to give a plausible operating point according to a set of measurements.

One of the assumptions for classical state estimation is that the topology of the system - defined by open or closed state of the breaker and switching devices - is already known. However, topology errors may occur and greatly jeopardize this process.

In order to face this problem, several approaches have been proposed. In this paper, two new topology estimation methodologies are introduced using a combination of neural networks and ITL optimization criteria. One of the objectives is to train such ANN structures to learn specific measurement manifolds in order to promptly and correctly predict the status of a given breaker or switching device. Moreover, it is desired to include infomax principles in the networks' training, seizing the maximum amount of information possible about the input data.

II. Autoencoders

Autoencoders are feedforward artificial neural networks trained with the objective of getting an output which mirrors its input with the minimum distortion possible. This kind of ANN may be viewed as the junction of two components: the first half - encoder - attempts to represent the input X in a space with less dimensions X' through the use of an encoding or compression function f. On the other hand, the second half - decoder - aims to do exactly the opposite, aiming to approximate f^{-1}. The typical structure of an autoencoder is to have an input layer, an hidden layer and an output layer.

Rui Oliveira (ruimmoliveira7@gmail.com) is with Faculty of Engineering of Universitu of Porto (FEUP), Portugal. Vladimiro Miranda (vmiranda@inesctec.pt) and Ricardo Bessa (rbessa@inesctec.pt) are with INESC TEC, Porto, Portugal.

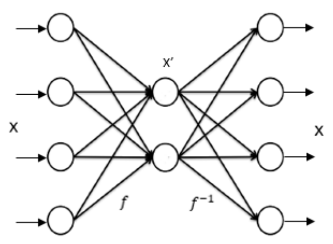

Fig. 1. Common architecture of an autoencoder.

III. ITL and Unsupervised Training

A. Renyi's Entropy

Alfred Renyi, willing to determine a more generalized definition of information content, proposed a parametric family of entropy functions, each one defined by a real parameter, α. So, for a probability distribution of a given random variable X, it is possible to express H_α as:

$$H_\alpha = \frac{1}{1-\alpha} \log \sum_{k=1}^{N} p_k^\alpha \quad with\, \alpha > 0,\ \alpha \neq 1 \tag{1}$$

Given its interesting characteristics, Renyi's quadratic entropy is frequently used in ITL. This particular variation of Renyi's entropy can be represented by:

$$H_2 = -\log \sum_{k=1}^{N} p_k^2 \tag{2}$$

Comparing Shannon's and Renyi's entropies, one can observe that the main difference is the placement of the logarithm in the expression: while Shannon's entropy is composed as a weighted sum of probability logarithms, Renyi's entropy is the logarithm of the sum of probabilities. These contrasts allow to infer that, for optimization algorithms, Renyi's entropy is a much more efficient metric than Shannon's expression.

B. The Maximum Entropy Criterion (MEC)

In machine learning, entropy can be specially appropriate in situations in which data representations are not precisely defined - such as unsupervised learning tasks - because this descriptor guides the algorithm through an infomax principle. Therefore, assuming a continuous random variable, y, being its pdf given by $f_Y(z)$, as well as combining both Renyi's quadratic entropy and Gaussian kernel pdf estimation, it is possible to construct an entropy estimator as expressed below:

$$H_2(y) = -\log \int_{-\infty}^{+\infty} f_y^2(z)\,dz \; = -\log\,V(y) \tag{3}$$

$$V(y) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(z-y_i,\sigma^2 I)\,G(z-y_j,\sigma^2 I)\,dz \tag{4}$$

V(y) is called the **information potential** and, through the convolution property of the Gaussian functions, can be written as:

$$V(y) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(y_i-y_j,2\sigma^2 I) \tag{5}$$

Attending to these last equations, one can state that if the information potential, V(y), is minimized, then the entropy, $H_2(y)$, will be maximized, assuring a maximum flow of information through the model. As a result, it is called the **Maximum Entropy Criterion** (MEC).

IV. Training Methodologies

Regarding the character of topology estimation using neural networks, it is possible to discriminate two possible alternatives: global and local topology estimation. Local topology estimation is a process that takes advantage of the remarkable local properties that analog measurements possess to determine open or closed state of a breaker. This is done by associating a single ANN or a set of ANN structures to every single breaker or switching device. This way, it is possible to integrate every single contribution to build larger, flexible models. In both training variants, a common principle of local topology estimation is followed.

A. *Data and Preprocessing*

As a test system, it was developed an IEEE RTS-24 variant to serve as the platform for the further studies with ANN structures. Some minor adjustments were made in order to further adapt the original case to the purposes of topology estimation, namely the introduction of breakers and analog measurement devices in buses and branches.

In order to obtain the database comprising all the studied cases, it was used a process very much similar to what is described in [1]. As a result, it was built a database with 11736 cases which were splitted under a 60%-20%-20% partition scheme for training, validation and test, respectively.

Before inputting these sets of raw data into the ANN structure, it was performed a pre-processing step which consisted in applying a normalization function that transformed every example - row - in an unit norm vector.

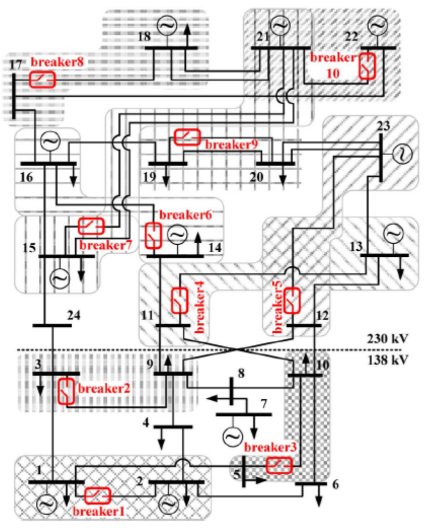

Fig. 2. IEEE RTS-24 Test Case.

B. *Deep AE*

This first training methodology is anchored on a competitive autoencoder principle to determine the breaker status similar to what has been described in [2] and [1]. Generally, this scheme assumes the existence of two autoencoders for every single breaker: one of the autoencoders is trained to learn open breaker scenarios while the other is responsible for learning the measurements pattern when the breaker is closed; the breaker status is given by the autoencoder with a lower error.

Regarding more technical aspects, the final model is constituted by two autoencoders, each one of them composed by an input layer with 18 neurons and having 5 hidden layers with sizes of 16, 14, 12, 14 and 16 neurons; the output layer has 18 inputs.

The weight starting point was defined through a random orthogonal initialization strategy described by Saxe et al. in [3]. The chosen optimization algorithm was ADADELTA; the existent optimization parameters were defined as recommended in [4].

The training procedure consists in an encoder layerwise training based on a MEC criterion being followed by a decoder training using a MSE cost function. The kernel size for entropy estimation through Parzen window was assigned as $\sigma = 0.4$. Finally, the training method used minibatches composed by 50 examples. The stopping criterion consists in early stopping through the evaluation of the model's performance on a validation set.

C. *Deep MLP*

The idea of adopting a multiple hidden layer MLP with a logistic regression layer is very tempting: in a first place, it serves the main purpose of local breaker status identification as a classification problem; the other major point is related to the reduction of ANN structures.

This training methodology has some common traits when contrasted to Deep AE, mainly regarding the optimization algorithm, the adopted kernel size and training mode. Thus, the used backpropagation algorithm was ADADELTA, the adopted kernel size was a constant σ of 0.4 and the model was trained using early stopping and minibatches composed by 50 examples.

The fundamental differences between Deep MLP and Deep AE lie in weight initialization and training procedure. Concretely, it was used a combination of random orthogonal for sigmoid layers and Xavier initialization for the last, logistic regression layer. At last, the training procedure consists in a layerwise pretraining through MEC and then a finetune was performed under the minimization of the negative log-likelihood.

V. Results

The models were developed in Python programming language, taking advantage of Theano and Lasagne deep learning libraries. The presented results for estimation of the breaker status of breaker 2 were obtained in a computer with an Intel Xeon E5-1260 v3 at 3.50 GHz CPU, with a 16.00 GB RAM and a Nvidia GeForce TITAN X GPU.

The subsequent tables summarize the relative performance of Deep AE and Deep MLP in terms of time and accuracy.

TABLE I. Confusion matrices for Deep AE and Deep MLP

	Deep AE	Deep MLP
True Closed States	1170	1247
False Closed States	5	0
True Open States	1161	1166
False Open States	78	1
Total Precision (%)	96.56	99.96

TABLE II. Running times for train and test of both models

	CPU (s)	CPU+GPU (s)	Acceleration GPU
Deep AE	85.638502	82.67714	1.04
Deep MLP	36.483892	21.214959	1.72

Observing both tables, it is easily noticeable the superior performance of Deep MLP over Deep AE, presenting only one misdiagnosed case against the 83 errors of Deep AE. The incorrect breaker status identification of Deep MLP was detected on the 2045th case of the test set and corresponds to a false open identification. Carrying out a manual verification of the active and reactive power flows in line 3-9, one verifies that those values are **-0.063302** and **-0.048085 p.u.**, correspondingly. In face of such reduced branch power flows, the wrong classification of Deep MLP in this particular scenario is understandable.

Apart from the error rates, it is necessary to pay close attention to temporal measurements. While in Deep AE the times for CPU and GPU are practically equal, the time needed to train and test the second methodology decreases to nearly a half when using GPU computing. Such differences in GPU acceleration are related to training procedures of both methodologies. In Deep AE, one autoencoder is trained after the other and the most prolonged task is the layerwise encoder training following a maximum entropy criterion. Given these two aspects, it is obvious that this solution offers almost no possibility of process parallelization due to its sequential character. On top of that, the part of the second methodology training which was more time-consuming was the generalized finetuning based on the minimization of the negative log-likelihood; this happens to be the part which benefits the most of GPU because of its capability of parallelizing calculations from different layers, resulting in reduced computational times.

Taking into account other similar works, it is also necessary to perform comparisons concerning those same approaches.

TABLE III. Highlight of accuracies of past published local topology estimation procedures.

Numerical Reference	Title of Article	Year of Publishing	Total Precision (%)
[2]	Towards an Auto-Associative Topology Estimator	2013	100.00
[1]	Breaker status uncovered by autoencoders under unsupervised maximum mutual information training	2014	99.86

In summary, it is easily perceived that, when contrasted to other previously published works, Deep AE has, unquestionably, a worse performance. On the other hand, Deep MLP has a higher successful identification ratio than in [1], although not presenting a 100% accuracy as exposed in [2]. However, in this latter article it is only adopted a competitive local autoencoder scheme along with a MSE-based training process, not implementing any ITL concept in autoencoder parameter optimization. As a result, one can infer that methodology 2, Deep MLP, has shown some progress towards an ITL-supported, neural network approach for topology estimation.

VI. Conclusion

The two main objectives have been achieved. On one hand, it has been proved that the combination of ITL cost functions along with the optimization of hyperparameters of both neural network models lead to breaker status predictions with great level of accuracy. On the other hand, noticeable time reductions in neural network training have been achieved through GPU computing, assessing the potential of this computational tool in the scope of a concrete power system application.

Unlike what has been affirmed in past works, it has been adopted a classic approach - Deep MLP - which does not need any additional external parameter definition regarding a splitting threshold between both status diagnosis.

References

[1] V. Miranda, J. Krstulovic, J. Hora, V. Palma, and J. C. Príncipe, "Breaker status uncovered by autoencoders under unsupervised maximum mutual information training," pp. 1–6.

[2] J. Krstulovic, V. Miranda, A. J. A. Simoes Costa, and J. Pereira, "Towards an auto-associative topology state estimator," *IEEE Transactions on Power Systems*, no. 3, pp. 3311–3318, aug.

[3] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," pp. 1–22, 2013. [Online]. Available: http://arxiv.org/abs/1312.6120

[4] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press.

[2] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks\r. *Science*, (5786):504–507. ,

[3] K Baldi, P & Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(10):53–58, 1998.

[4] Andrea Basso. Autoassociative neural networks for image compression: a massively parallel implementation. *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 373–381.

[5] B Golomb and T Sejnowski. Sex recognition from faces using neural networks. *in A. Murray (ed.), in Applications for Neural Networks*, pages 71–92.

[6] S. Narayanan, R.J. Marks, J.L. Vian, J.J. Choi, M.A. El-Sharkawi, and B.B. Thompson. Set constraint discovery: missing sensor data restoration using autoassociative regression machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, pages 2872–2877. IEEE.

[7] Pascal Vincent PASCALVINCENT and Hugo Larochelle LAROCHEH. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol. *Journal of Machine Learning Research*, 11:3371–3408, 2010. ,

[8] 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. ,

[9] Salah Rifai and Xavier Muller. Contractive Auto-Encoders : Explicit Invariance During Feature Extraction. *Icml*, (1):833–840. ,

[10] Carl Doersch. Tutorial on Variational Autoencoders. *arXiv*, pages 1–23, jun.

[11] Telmo Amaral, Luís M Silva, Luís A Alexandre, Chetak Kandaswamy, Jorge M Santos, and Joaquim Marques De Sá. Using Different Cost Functions to Train Stacked Auto-encoders.

[12] Vladimiro Miranda, Jakov Krstulovic, Joana Hora, Vera Palma, and José C Príncipe. Breaker status uncovered by autoencoders under unsupervised maximum mutual information training. pages 1–6.

[13] Jose C Principe. *Information Theoretic Learning*. Number XIV. ,

[14] G Deco and D Obradovic. *An Information-Theoretic Approach to Neural Computing*. Perspectives in Neural Computing. Springer New York, New York, NY.

[15] Martin T. Hagan and Mohammad B. Menhaj. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.

[16] E. M. Johansson, U. Dowla, and D. M. Goodman. Backpropagation learning for multilayer feed-forward neural network using the conjugate gradient method. *International Journal of Neural Systems*, 2(4):291–301, 1992. ,

[17] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *IEEE International Conference on Neural Networks - Conference Proceedings*, 1993-January:586–591, 1993.

[18] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, (7):1527–54. ,

[19] James Martens. Deep learning via Hessian-free optimization. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742.

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

[21] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 315–323. ,

[22] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. Neural Networks: Tricks of the Trade. pages 9–50. Springer Berlin Heidelberg.

[23] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, pages 1–22.

[24] Mathias Seuret, Michele Alberti, Rolf Ingold, and Marcus Liwicki. PCA-Initialized Deep Neural Networks Applied To Document Image Analysis.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, feb. ,

[26] F C Schweppe and J Wildes. Power system static state estimation. I - exact model. In *1969 power industry computer applications conference, 18-21 May 1969*, pages 85–91, 1969.

[27] F C Schweppe and D Rom. Power system static state estimation. II-approximate model. In *1969 power industry computer applications conference, 18-21 May 1969*, pages 92–99, 1969.

[28] F C Schweppe. Power system static state estimation. III - implementation. In *1969 power industry computer applications conference, 18-21 May 1969*, pages 100–106, 1969.

[29] A. Monticelli. *State Estimation in Electric Power Systems*. Springer US, Boston, MA.

[30] Ali Abur and Antonio. Gomez Exposito. *Power system state estimation : theory and implementation*. Marcel Dekker, 2004.

[31] Yousu Chen PNNL. Weighted-Least-Square (WLS) State Estimation. 2015.

[32] A. Simoes Costa and J. A. Leao. Identification of topology errors in power system state estimation. *IEEE Transactions on Power Systems*, 8(4):1531–1538, 1993.

[33] K. A. Clements and P. W. Davis. Detection and identification of topology errors in electric power systems. *IEEE Transactions on Power Systems*, 3(4):1748–1753, 1988.

[34] Felix F. Wu and Wen Hsiung E Liu. Detection of topology errors by state estimation. *IEEE Power Engineering Review*, 9(2):50–51, 1989.

[35] R. Lugtu, D. Hackett, K. Liu, and D. Might. Power System State Estimation: Detection of Topological Errors. *IEEE Transactions on Power Apparatus and Systems*, PAS-99(6):2406–2412, 1980.

[36] M.R. Irving and M.J.H. Sterling. Substation data validation. *IEE Proceedings C Generation, Transmission and Distribution*, 129(3):119, 1982.

[37] N. Singh and H. Glavitsch. Detection And Identification Of Topological Errors In Online Power System Analysis. *IEEE Transactions on Power Systems*, 6(1):324–331, 1991.

[38] O. Alsac, N. Vempati, B. Stott, and A. Monticelli. Generalized state estimation. *IEEE Transactions on Power Systems*, (3):1069–1075.

[39] A. Monticelli and A. Garcia. Modeling zero impedance branches in power system state estimation. *IEEE Transactions on Power Systems*, (4):1561–1570.

[40] A. Monticelli. Modeling circuit breakers in weighted least squares state estimation. *IEEE Transactions on Power Systems*, 8(3):1143–1149, 1993.

[41] Kevin A. Clements. Topology error identification using normalized lagrange multipliers. *IEEE Transactions on Power Systems*, (2):347–353, may.

[42] Elizete Maria Lourenço, Antonio Simões Costa, and Kevin A. Clements. Bayesian-based hypothesis testing for topology error identification in generalized state estimation. *IEEE Transactions on Power Systems*, 19(2):1206–1215, 2004.

[43] Elizete M. Lourenco, Antonio J. A. Simoes Costa, Kenvin A. Clements, and Rafael A. Cernev. A topology error identification method directly based on collinearity tests. In *2005 IEEE Russia Power Tech*, pages 1–6. IEEE, jun.

[44] Jorge Pereira, Vladimiro Miranda, and J. Tomé Saraiva. Fuzzy control of state estimation robustness. *Proceedings of the 14th PSCC*, (June):24–28, 2002.

[45] N. Vempati, C. Silva, O. Alsac, and B. Stott. Topology estimation. In *IEEE Power Engineering Society General Meeting, 2005*, pages 2069–2073. IEEE.

[46] Eduardo Caro, Antonio J. Conejo, and Ali Abur. Breaker status identification. *IEEE Transactions on Power Systems*, 25(2):694–702, 2010.

[47] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang. A pattern analysis approach for topology determination, bad data correction and missing measurement estimation in power systems. *Power Symposium, 1990. Proceedings of the Twenty-Second Annual North American*, pages 363 – 372.

[48] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang. Neural networks for topology determination of power systems. *Proceedings of the First International Forum on Applications of Neural Networks to Power Systems*, pages 297–301.

[49] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang. Solving data acquisition and processing problems in power systems using a pattern analysis approach. *IEEE Proceedings-Generation, Transmission and Distribution*, (4):365–376.

[50] J C S Souza, A M da Silva, and A P da Silva. Online topology determination and bad data suppression in power system operation using artificial neural networks. In *20th International Conference on Power Industry Computer Applications., 1997*, pages 46–53. IEEE.

[51] D.M. Vinod Kumar, S.C. Srivastava, S. Shah, and S. Mathur. Topology processing and static state estimation using artificial neural networks. *IEE Proceedings - Generation, Transmission and Distribution*, (1):99.

[52] D. Singh, J. P. Pandey, and D. S. Chauhan. Topology identification, bad data processing, and state estimation using fuzzy pattern matching. *IEEE Transactions on Power Systems*, 20(3):1570–1579, 2005.

[53] Vladimiro Miranda, Jakov Krstulovic, Hrvoje Keko, Cristiano Moreira, and Jorge Pereira. Reconstructing missing data in state estimation with autoencoders. *IEEE Transactions on Power Systems*, 27(2):604–611, 2012.

[54] Jakov Krstulovic, Vladimiro Miranda, Antonio J. A. Simoes Costa, and Jorge Pereira. Towards an auto-associative topology state estimator. *IEEE Transactions on Power Systems*, (3):3311–3318, aug.

[55] P.N. Pereira Barbeiro, J. Krstulovic, H. Teixeira, J. Pereira, F.J. Soares, and J.P. Iria. State estimation in distribution smart grids using autoencoders. *Proceedings of the 2014 IEEE 8th International Power Engineering and Optimization Conference, PEOCO 2014*, (March):358–363, 2014.

[56] B.B. Thompson, R.J. Marks, J.J. Choi, M.A. El-Sharkawi, and C. Bunje. Implicit learning in autoencoder novelty assessment. *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*, pages 2878–2883.

[57] Marco Martinelli, Enrico Tronci, Giovanni Dipoppa, and Claudio Balducelli. Electric Power System Anomaly Detection Using Neural Networks. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 1242—-1248, 2004.

[58] Salman Mohagheghi, Ganesh K. Venayagamoorthy, and Ronald G. Harley. Optimal wide area controller and state predictor for a power system. *IEEE Transactions on Power Systems*, 22(2):693–705, 2007.

[59] Wei Qiao, Zhi Gao, Ronald G. Harley, and Ganesh K. Venayagamoorthy. Robust neuro-identification of nonlinear plants in electric power systems with missing sensor measurements. *Engineering Applications of Artificial Intelligence*, 21(4):604–618, 2008.

[60] Vladimiro Miranda, Adriana R Garcez Castro, and Shigeaki Lima. Diagnosing faults in power transformers with autoassociative neural networks and mean shift. *IEEE Transactions on Power Delivery*, 27(3):1350–1357, 2012.

[61] Kunjin Chen, Jun Hu, and Jinliang He. A Framework for Automatically Extracting Overvoltage Features Based on Sparse Autoencoder. *IEEE Transactions on Smart Grid*, (c):1–1.

[62] Long Wang, Zijun Zhang, Jia Xu, and Ruihua Liu. Wind Turbine Blade Breakage Monitoring with Deep Autoencoders. *IEEE Transactions on Smart Grid*, (c):1–1.

[63] Vladimiro Miranda, Joana Da Hora Martins, and Vera Palma. Optimizing large scale problems with metaheuristics in a reduced space mapped by autoencoders-application to the wind-hydro coordination. *IEEE Transactions on Power Systems*, 29(6):3078–3085, 2014.

[64] Jack Kelly and William Knottenbelt. Neural NILM: Deep Neural Networks Applied to Energy Disaggregation. jul. ,

[65] Qinghua Hu, Rujia Zhang, and Yucan Zhou. Transfer learning for short-term wind speed prediction with deep neural networks. *Renewable Energy*, pages 83–95.

[66] Claude E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, (July 1928):379–423. ,

[67] D. Erdogmus and J.C. Principe. Entropy minimization algorithm for multilayer perceptrons. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, pages 3003–3008. IEEE.

[68] Probability Subcommittee. IEEE Reliability Test System. *IEEE Transactions on Power Apparatus and Systems*, (6):2047–2054.

[69] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method.