Learning and Nonlinear Models - Journal of the Brazilian Society on Computational Intelligence (SBIC), Vol. 18, Iss. 2, pp. 40-55, 2020

EVOLVING DEEP NEURAL NETWORKS FOR TIME SERIES FORECASTING

Lídio Mauro Lima de Campos, Jherson Haryson Almeida Pereira, Danilo Souza Duarte
Faculdade de Computacão - ICEN, Universidade Federal do Pará
lidio@ufpa.br, harysonjherson@gmail.com, danilo.duarte@hotmail.com
Orcid iD: https://orcid.org/0000-0003-4315-829X

Roberto Célio Limão de Oliveira
Faculdade de Eng. da Computacão - ITEC, Universidade Federal do Pará
limao@ufpa.br

Resumo – O objetivo deste artigo é introduzir uma abordagem inspirada biologicamente que possa gerar automaticamente redes neurais profundas com boa capacidade de previsão, menor erro e grande tolerância a ruídos. Para tanto, foram utilizados três paradigmas biológicos: Algoritmo Genético (GA), Sistema Lindenmayer e Redes Neurais. As seções finais do artigo apresentam alguns experimentos com o objetivo de investigar as possibilidades do método na previsão do preço da energia no mercado brasileiro. O modelo proposto considera uma previsão de preço em várias etapas (12, 24 e 36 semanas á frente). Os resultados para redes MLP e LSTM mostram boa capacidade de prever picos e precisão satisfatória de acordo com medidas de erro comparadas com outros métodos.

Palavras-chave – NeuroEvolução, Previsão de séries temporais; redes neurais recorrentes, redes LSTM.

Abstract –The aim of this paper is to introduce a biologically inspired approach that can automatically generate Deep Neural networks with good prediction capacity, smaller error and large tolerance to noises. In order to do this, three biological paradims were used: Genetic Algorithm (GA), Lindenmayer System and Neural Networks (DNNs). The final sections of the paper presents some experiments aimed at investigating the possibilities of the method in forecast the price of energy in the Brazilian market. The proposed model considers a multi-step ahead price prediction (12, 24, and 36 weeks-ahead). The results for MLP and LSTM networks show good ability to predict peaks and satisfactory accuracy according to error measures comparing with other methods.

Keywords – NeuroEvolution; Time series forecasting; recurrent neural networks, LSTM networks.

1. INTRODUCTION

During the 1990s, the Brazilian electricity regulatory model began to be restructured to promote competition in the context of generating and trading activities, as well as the construction of efficient regulation for transmission and distribution activities, as well as to attract investment to the sector. The energy market reform has brought greater risks associated with energy procurement. The how much? and when? to buy on the spot market are crucial decisions for power companies. Because such decisions depend on the price of electricity, a misguided investment strategy today could cost millions of dollars in the future.

Forecasting the price of electricity is an important issue for all market participants to decide on the most appropriate bidding strategies and to establish bilateral contracts that maximize their profits and minimize their risks. Currently some forecast methods have been proposed in the literature [1], [2], [3], [4].

A neuro-evolutive algorithm (NEA) is among the evolutionary algorithms (EAs) used in the design and/or training of an artificial neural network (ANN). Bio-inspired Algorithms (BIOAs) have gained popularity due to their efficiency at solving different non-linear optimization problems [5]. In this context, this paper tests the possibilities of a hybrid system called Artificial Development and Evolution of Deep Neural Networks (ADEANN-Deep) for shor-term energy price prediction using explanatory variables.

Our approach is inspired by two natural biological mechanisms: genetic encoding and the evolution of genetic coding. As is well-known, neuron development is governed by the genetic information encoded in deoxyribonucleic acid (DNA), ultimately generating the final shape of the brain. During biological development, in the complex process that links the DNA code to its phenotype, the same gene is used in many contexts. This compactness minimizes the information required to describe complex individuals. In contrast, evolution describes the temporal changes in the genetic code (DNA). Among the several mechanisms underlying these evolutionary changes, natural selection is very important.

The two natural processes described above are hybridized such that the DNA contained in cells can also spawn cells, while the changes in DNA are passed onto later generations. Motivated by these natural processes, we propose an artificial hybrid system that abstracts these natural mechanisms at an acceptable level of complexity.

To this end, we propose a biologically inspired artificial hybrid system called Artificial Development and Evolution of Deep ANNs (ADEANN-Deep). The ADEANN-Deep integrates two components, the first is a generative representation that represents

genotypes (a set of production rules of a Lindenmayer system) by a compact IES. To mimic the DNA encoding scheme and enable scalability, our IES leverages the phenotype representation to a smaller genotype. Thus, the search process is carried out in a lower-dimensional solution space. The second component is a genetic algorithm (GA), a simplified representation of natural evolution. In local search problems based on GAs, a bit string is called a chromosome (the genotype). Each bit on the chromosome is a gene, and a gene set represents the parameters of a function to be optimized. Each string is assigned with a fitness that indicates the quality of its encoded solution (the phenotype). To improve its biological realism, the GA in our approach evolves the generative representation. The evolutionary process can be regarded as the temporal genetic changes in the hypothetical DNAs of a population of individuals, regulated by an artificial selection mechanism. The above biological inspiration underlies the originality of our approach. The general structure of ADEANN-Deep is shown in Figure 1. The hybrid system is described in detail in section 4.

The main contribution of our method is the genotype representation by our proposed IES. Using a compact DNA encoding, we codify a parametric Lindenmayer system (L-system) with memory, which implements the principles of organization, modularity, repetition and hierarchy to achieve complex neural architectures (multilayer and recurrent networks). [6] and [7] adopted L-systems. Despite of using DNA encoding, their study was restricted to feedforward neural networks, whereas our approach is extended to recurrent networks, such as LSTM networks and other topologies, including deep feed forward and radial basis network. In the IES used by [6], the genotypes encode twenty rewrite rules of an L-system. Our DNA encoding system encodes a parametric L-system with memory using 10 production rules. Therefore, our IES is more compact than the method proposed by [6] and reduces the search space of all feasible solutions. In addition, the memory mechanism in our approach enables the reuse of phenotypic structures (rewrite of nodes and connections) at different stages of development. Such reuse is an important capability of NEAs. In other words, at the moment, the greatest contribution of our approach is methodological, since ADEANN-Deep allows us to evolve recurrent neural networks, which allow us to simulate a larger class of problems, such as dynamical systems.

The relevance of our approach relies in the proposal that brains manipulate recurrent networks that are capable of forming more complex computation than that present in feedforward neural networks. As it can be difficult for recurrent networks to learn long sequences, our results are limited by computational power. Thus, a way to train neural networks in a GPU architecture should be investigated in order to provide a significant computation speed-up over CPU-only training.

The first version of ADEANN was totally developed in C language. In addition, the system only allowed to evolve direct and recurrent neural networks with three layers. In relation to the previous version of ADEANN [8], this research presents the following improvements: system migration to Python language justified by its applicability and portability in the artificial intelligence area, which enabled integration with prominent frameworks used in the current market for data processing, like Pandas and data science, like Keras and Tensorflow. This new version of the hybrid system (ADEANN-Deep) using Keras / Tensorflow has expanded the possibility of the system to use various deep and recurrent Neural Network architectures. However, the system is still being improved and is expected to become a tool for training and validation of deep neural networks. Thus, the hybrid system will offer great technical contributions.

This paper is organized as follows. Section 2 presents the background, section 2.1 dicusses related work, section 2.2 presents the features of the Brazilian electricity market, section 2.3 introduces the neural network architectures used in this research. Section 3 describes a new approach to formalize the problem of ADANNs (artificial development and evolution of ANNs) as a local search based on rational agents. Section 4 introduces a biologically inspired method for automatic design of ANNs. Section 5 presents Material and Methods. Lastly, simulation results and conclusions are presented in Sections 6 and 7, respectively.

2. BACKGROUND

2.1. RELATED WORK

The paper [9], proposes a new hybrid approach for short-term energy price prediction. The approach combines auto-regressive moving average (ARIMA) and neural network (NN) models in a cascaded structure and uses explanatory variables. A two step procedure is applied. In the first step, the selected explanatory variables are predicted, in the second one, the energy prices are forecasted by using the explanatory variables prediction. Nevertheless, the authors restricted their approach to Multilayer Perceptron (MLP), which is unsuitable to simulate temporal series forecasting.

[10] proposes an automated method, CoDeepNEAT, for optimizing deep learning architectures through evolution. By extending existing neuroevolution methods to topology, components and hyperparameters, the method achieves results comparable to best human designs in standard benchmarks in object recognition and language modeling. It also supports building a real-world application of automated image captioning on a magazine website. Given the anticipated increase in available computing power, evolution of deep networks is promising approach to constructing deep learning applications in the future. The results are limited by the available computational power. It would be required to train networks only partially during evolution.

The paper [11], proposes a new prospect for evolving optimized Deep Neural Networks (DNNs) which can provide a warm start to the training process compared to heuristic random initial architecture. They discuss the theoretical approach towards possibility of optimizing the learning process inspired from the existing un-conventional approaches. The training process of DNN with Evolutionary Computation(EC) approach is faster than regular approach by a considerable difference of over 6 hours for MNIST data set. Further, they observed a considerable improvement in the classification accuracies. A challenge to be overcome by this method is to decide the initial topology, weights and biases to start with.

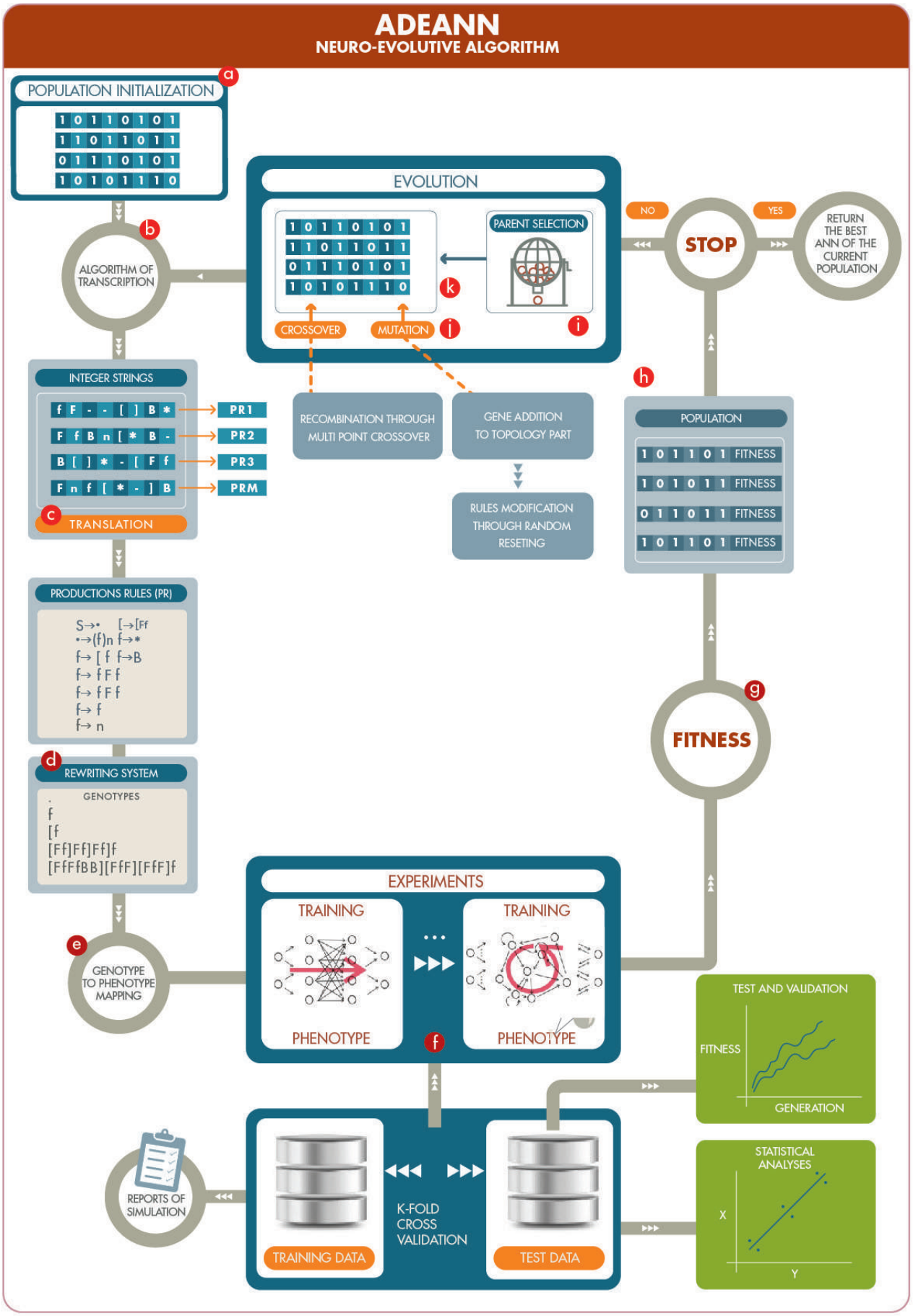


Figure 1: The general structure of ADEANN-Deep.

The work [12], provides a qualitative evaluation of four of the most popular Deep Learning frameworks, including: Caffe, Torch, Lasagne and TensorFlow. A printed character recognition task was used as case study and a Convolutional Neural Network was implemented for this purpose. The analysis focus on issues that are important for the development process and encompasses nine qualitative dimensions, showing the strengths and weaknesses of each framework. During the development of this work, some difficulties were found when using the frameworks for the specific application previously described. Problems with GPU memory allocation appeared in both Torch and TensorFlow.

The study [13] aims at filling in the knowledge gap of deep learning-based techniques for day-ahead multi-step load forecasting in commercial buildings. Two classical deep neural network models, namely recurrent neural network(RNN) and convolutional neural network (CNN) have been proposed and formulated under both recursive and direct multi-step manners. Their performances are compared with the Seasonal ARIMAX model with regard to accuracy, computational efficiency, generalizability and robustness. Among all of the investigated deep learning techniques, the gated 24h CNN model, performed in a direct multi-step manner, proves itself to have the best performance,improving the forecasting accuracy by 22.6% compared to that of the seasonal ARIMAX. A limitation of the method is that neural network architecture is designed through heuristic methods.

[14] used a Hybrid neural model (HIRA model) for Short-Term Electricity price forecasting, which was also applied and tested in the scope of two eletricity markets: German and Hungarian. In Hungary, thermal plants have 90% of installed capacity and represent a stable source of electricity, but generally involving a large installed capacity per unit. Hungary uses bid-based market by calculating the electricity price forescast at three levels: Hourly Forecasting error, Peak load Forecasting Error, and Base Load Forecasting Error.

2.2 BRAZILIAN ELECTRICITY MARKET

The Brazilian energy market operates with two trading environments, one regulated and the other free. The former involves a pool of purchasing agents buying power from selling agents (generators, independent power producers or self-producers) in public auctions under set prices, while in the latter market buyers and sellers are free to establish bilateral contracts and negociate prices and conditions [15]. The difference between the quantity of energy contracted and that effectively consumed or produced by the agents is accounted in the short-term market based on the spot price called PLD (settlement price for the differences) [9]. PLD is calculated weekly and it is based on the system marginal cost of operation obtained from an optimization process to dispatch generators. The PLD is established by the Brazilian Electricity Regulatory Agency (ANEEL) and is evaluated to each submarket associated with the country regions: North, Northeast, Center-west/Southeast, and South.

Brazil uses a cost-based market instead of a bid-based market, and adopts a tight pool model with a centralized and least cost dispatch organized by National System Operator (ONS). This scheme is adopted due the country peculiarities, which has an installed capacity of 121 GW where 65.96 % corresponds to hydro generation. The hydro system is composed of several reservoirs capable of multi-year regulation located at the same river with differents owners [9].

2.3 NEURAL NETWORK

Neural Networks have beeen successfully aplied to a variety of complex problems due to its ability to learn non-linear relationships between input and output patterns, which would be difficult to model conventional methods [9]. In this research, ADEANN-Deep enables automatic design of different recurrent and deep neural network architectures that yields the best generalization accuracy for each submarket. The neural network architectures used in this work are presented in the next subsections.

2.3.1 MULTILAYER PERCEPTRON - MLP

The first network model generated by ADEANN in the classification experiment was a single hidden-layer feed-forward network. This model, characterized by a three-layer network of simple processing units connected by acyclic links (Figure 2), is commonly employed in systems biology studies, including evolutionary studies. Information flows through the ANN in discrete time. The output o_j of node j is calculated by equation 1.

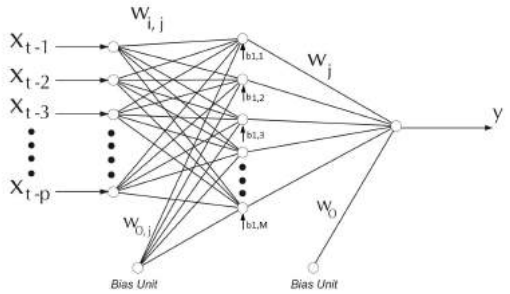


Figure 2: Structure of Multilayer Perceptron

$$\vartheta(o_j) = \frac{1}{1 + exp^{-k.\sum_{i \epsilon I_j}(w_{ij}x_i + b_j)}} \tag{1}$$

Here, I_j is the set of nodes connected to node j, w_{ij} is the strength of the connection between node i and node j, o is the output value of node i, and b_j is the bias. The parameter k measures the steepness of the sigmoidal function (equation 1). As k is positive, the sigmoidal function is monotonically increasing, continuous and differentiable over the whole domain. In our experiments, we assigned the typical value for the training of neural networks with BP (namely, k = 1). The parameters of the neural network w_{ij} are changed by an amount Δw_{ij}, which is calculated by equation 2.

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{2}$$

where the parameter η is the learning rate and *E* is the error in the output layer. The δ term in equation 3 is a momentum term, to accelerate the learning process while avoiding instability in the algorithm.

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \delta \Delta w_{ij}(t) \tag{3}$$

2.3.2 LONG SHORT-TERM MEMORY-LSTM

The Recurrent neural network dynamics can be formulated by deterministic transitions from previous to current hidden states. The deterministic state transition is described in equation 4, the architecture of the LSTM neural network is shown in Figure 3. A typical LSTM network is compoosed of memory cells. The cell state and the hidden state are transferred to the next cell. A gate is analogous to a layer or a chain of matrix operations carrying discrete weights. LSTMs are projected to prevent the long-term dependency problem for using gates to regulate the memorizing process.

$$RNN : h_t^{l-1}, h_{t-1}^l \rightarrow h_t^l \tag{4}$$

The first stage in constructing an LSTM network is to determinte information that is not required and omit it from the cell at that stage. For such purpose, it uses a sigmoid activation function, which takes the output of the last LSTM unit (h_{t-1}) at time $t-1$ and the current input (X_t) at time t. This gate is called the forget gate (f_t), which is a vector with values ranging from 0 to 1, corresponding to each number in the cell state C_{t-1}.

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f) \tag{5}$$

where: σ is the sigmoid function and W_f and b_f are the weights matrix and bias of f_t.

The next step is to decide which new information will be stored in the state of the cell. Firstly, another gate containing a sigmoid, known as an input layer decides which values will be updated (i_t.). Then, a layer with a hyperbolic tangent (tanh) creates a vector of new candidate values (C_{t-1}) that can be added in the state of the cell. Subsequently, these two outputs are combined to generate an update of the cell state.

The next step is to decide which new information will be stored in the state of the cell. Firstly, the sigmoid layer, known as an input layer, decides whether the new information should be updated or ignored i_t. Then, a layer with a hyperbolic tangent (tanh) gives weight to the values which passed by, deciding their level of importance (-1 to 1). Subsequently, these two outputs are multiplied to generate an update of the cell state. This new memory is then added to old memory C_{t-1} resulting in C_t.

$$i_t = \sigma(W_i.[h_{t-1}, X_t] + b_i) \tag{6}$$

$$M_t = tanh(W_m.[h_{t-1}, X_t] + b_m) \tag{7}$$

$$C_t = f_t.C_{t-1} + i_t.M_t \tag{8}$$

where:C_{t-1} and C_t are the cell states at time t-1 and t, while W and b are the weight matrices and bias of the cell state.

Finally, the LSTM output (h_t) is calculated based on the state of the filtered cell (o_t). Firstly, a sigmoid is calculated to decide which parts of the cell state reach the output. Subsequently, the output of sigmoide gate (o_t) is multiplied by the new values created by the tanh layer from the cell state (C_t), with a value ranging between -1 and 1.

where: Wo and bo are the weight matrices and bias of the output gate.

$$o_t = \sigma(W_o.[h_{t-1}, X_t] + b_o) \tag{9}$$

$$h_t = o_t.tanh(C_t) \tag{10}$$

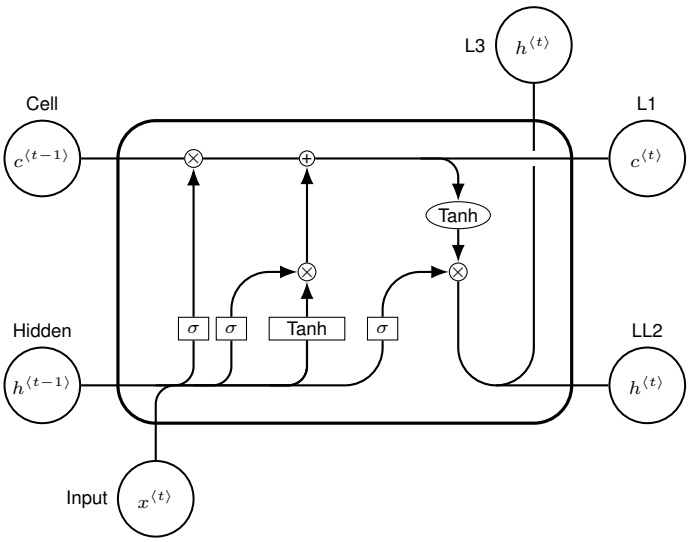


Figure 3: The structure of the Long Short-Term Memory (LSTM) neural network. Reproduced from [16]

3 OUTLINE OF THE APPROACH

Our approach involves the formulation of an artificial neural network design as an optimization problem (ANNDP), [17] that is: given a set of L observations on the behavior of a particular process, Ψ=$\{(xd^l, yd^l)\}$, l = 1...L , where xd^l represents a numeric vector defined in R^n and yd^l is a numeric vector defined in R^m, the goal is to find an ANN's topology, yc^l = ANN(w*, xd^l), which minimizes the mean square error between yd^l and yc^l, this is, between the desired values in the observations set and the computed values in the neurons' outputs situated in the ANN's output layer.

An ANN topology can be described as a finite set of neurons, that is, nodes of an oriented graph Nodes =$\{$n1, n2, ..nk$\}$, and a finite set H $\subseteq$ N x N of connections between neurons, which means directed edges in graphs notation. An input layer is a set of input units, that is, a subset of n nodes whereas an output layer is a set of output units, namely a subset of m nodes. In feed-forward ANNs(FANNs), the k^{th} layer (k $>$ 1) is the set of all nodes n_i ϵ Nodes . These types of nodes have an edge path of length k-1 between some input unit and u. In fully connected recurrent ANNs (RANNs), all units have connections to all non-input units.

We approach the solution of ANNDP based on the methodology of Russell and Norvig [18] called problem-solving-agent, whose agent is named ADEANN (Artificial Development and Evolution of ANNs), which encapsulates a special scheme of solutions representation as well as a local search strategy based on genetic algorithms to solve the problem. Regarding the representation scheme, the approach adopts a generative representation, which means that, instead of an encoded ANN topology, each chromosome stores a set of production rules of a Lindenmayer system (Figure 1 a,k,h) which, in turn, generates ANN's, (Figure1 f), topologies regarding the solution process, the SEARCH-ANN function outlined below illustrates the structure of the program in the ADEANN agent.

Function SEARCH−ANN(SearchParam, TransitionModel, FitnessFunction) **return** an ANN topology

---

1:inputs: SearchParam, TransitionModel, FitnessFunction
2:vars: Pop, t, PopPerformances;
3:k← 0
4:$ANNsPop^k$←Generate-ANNs(SearchParam)
5:$ANNsPerformance$← Evaluate-ANNs($ANNsPop^k$, FitnessFunction)
6:loop do
7:if StopConditionTest(k, ANNsPerformance, SearchParam)
8:then return solution(Best-ANN($ANNsPop^k$, ANNsPerformance)
9:$ANNsPop^{(k+1)}$←($ANNsPop^{(k+1)}$, ANNsPerformance, TransitionModel, SearchParam)
10:ANNsPerformance←Evaluate-ANNs($ANNsPop^{(k+1)}$, FitnessFunction)
11:k ←k+1
12:end

Rule Identifier	Rule
1,2	S→**.** (axiom) (2) .→(f...f)n
3	(3.1) f→[f (3.2) f→fFf (3.3) f→fF (3.4) f→n
3,4,5	(3.5) f→f (3.6) f→fB (4) [→[Ff] (5) f→f*

Table 1: **The production rules of the parametric L-System with memory**

The SEARCH-ANN function starts the local search process aiming at achieving an artificial neural network topology yc^l, yc^l=ANN$\{(w^*, xd^l)\}$, which minimizes the mean square error between yd^l and yc^l, for l = 1...L in the ANNDP's formulation (stop condition in Figure 1). This function employs information on the search parameters (SearchParam input term) as well as a transition model (TransitionModel input term) to describe how to modify current populations of ANNs and generate a new population, (Figures 1 i,j,k), in addition to an evaluation function (FitnessFunction input term), (Figure 1 g), to measure the value of each ANN in a current population.

Firstly, in the beginning of the process, Generate-ANNs function generates an initial population of ANNs, (Figure 1 a), in which each ANN is represented by a set of production rules codified in a chromosome (bit values 0 and 1). This function considers the information in the SearchParam input term on the desired number of ANNs in the populations as well as on the desired length for the chromosomes in the population. Evaluate-ANNs function stores in ANNsPerformace the computed performance value of each ANN topology in the current population based on the mean square error computed in the output layer of the ANN-SEARCH-ANN function, (Figure 1 g), which employs an iteration counter (k) and a condition named StopConditionTest boolean function to decide when to stop the local search process and return a solution to a problem (stop condition in Figure 1). The description of the stop condition is based on a proposition relating the information on the current iteration counter k and the information available in the SearchParam input term. This means that the max number of loops in its repetition scheme is central to the local search strategy in the approach, as well as an ideal performance value such that for an ANN to be considered a solution. Modify-ANN function is executed repeatedly seeking to transform a current population of ANNs in a new population of ANNs (Figures 1 i,j,k). In our approach, this function encapsulates the evolutionary principles of pairs selection and crossing over pairs and individual mutation (Figure 1 j) . Central to the approach, compact indirect encoding scheme (IES) conducts and controls the process of mapping a set of production rules of a Lindenmayer system, (Figure 1 c), codified in a chromosome to an associated ANN topology (Figure 1 e).

4 BIOLOGICALLY INSPIRED NEA

The optimization process of ADEANN-Deep proceeds through several stages. The GA starts with a population of individuals randomly initialized with 0s and 1s (Figure 1 a). Second , the bits of each individual of the population are subjected to transcription (Figure 1 b) and translation (Figure 1 c), following valid production rules. After finding the appropriate production rules (Table 1), the rewriting system generates the genotypes (Figure 1 d). All of the genotypes are mapped to phenotypes (ANN architectures) (Figure 1 e). The ANNs are then trained (Figure 1 f) and validated and tests are carried out. The performance in predicition task of each ANN is measured from its fitness (Figure 1 g). The genotypes are classified by the performances of their ANNs (Figure 1 h). The GA selects the best individuals (Figure1 i) for mutation and crossover operations (Figure 1 j), which provide the new population (Figure 1 k). The previous steps are repeated through n generations. The following subsections describe the three subsystems of ADEANN-Deep.

4.1 L-SYSTEM BASED ARTIFICIAL EMBRYOGENESIS MODEL

To mimic the mechanism of grown structures, including neurons, we adopt a parametric L-system with memory. It comprises a set of rules created from an alphabet. This system can be described as a grammar $G = \{\Sigma, \Pi, \alpha\}$, where the alphabet consists of the elements of the set $\Sigma = \{., f, F, n, [,], *, B\}$ and the production rules (Π) described in Table 1. The axiom α = . is the starting point of the developmental process, where f denotes a neuron and F is a connection between neurons, [and] indicate storage and recovery, respectively, of the current state of the development, * denotes that the string is recovered from storage and B is the connection of a neuron with a block of neurons. The second rule .→(f...f)n, means replace the start point by the neurons of the input layer. Rule 3.1 (f→[f) means to store the position of the current neuron, so as to start a new ramification from it. Rule 3.2 (f→fFf) means establish a connection between two neurons. Rule 3.3 (f→fF) means establishing a connection from a specific neuron. Rule 3.4 (f→n) means replace a provisional neuron with a permanent neuron. Rule 3.5 (f→f) means to maintain a specific neuron during development. Rule 3.6 (f→fB) means connect a neuron to a block of neurons. Rule 4 ([→[Ff]) means start the development of a new ramification from a specific neuron and recover the previous state. Rule 5(f→f*) means recover a previous ramification stored for use.

As a simple example, suppose that starting with the axiom (α = .) twice , and applying the second production rule . → f to the axiom twice, the resulting string is **ff**. Applying the third rule (3.1) $f \rightarrow [f$ to string **ff** yields a new string, **f[f]**. After one, two, three and applications of the fourth rule $[\rightarrow [Ff]$ on string **f[f**, the string becomes **[Ff]Ff]f[Ff]Ff]f**. Applying the rule (3.3) $f \rightarrow$**fFf** eight time to the previous string, the resulting string is **[FfFfFf]FfFfFf]f [FfFfFf]FfFfFf]f** . After Applying the (3.6) $f \rightarrow$**fB** to the previous string the resulting string is **[FfBFfBFfB]FfBFfBFfB]f [FfBFfBFfB]FfBFfBFfB]f**. Finally applying the rule (3.4) $f \rightarrow$**n** to the previous string the resulting string is **[FfnFfnFfn]FfnFfnFfn]f [FfBFfBFfB]FfBFfBFfB]f**. This phenotype represents the RNA structure shown in Figure 4.

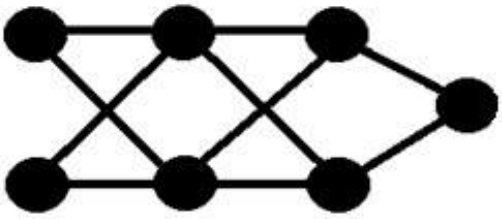


Figure 4: A simple example of the construction process of a branch of an iterated ANN using the rules of the L-system is illustrated in Table 1.

4.2 RULE EXTRACTION BY GENETIC ALGORITHMS

The neurons generated in the previous subsection are developed after the following process. To formulate a biologically realistic GA, we let the genes of the chromosomes (sequences of hypothetical DNA) encode a recipe (the production rules of the L-system described in subsection 4.1 and illustrated in Table 1. The recursive rules in Table 1 drive the developmental stages of the neurons (Figure 4).

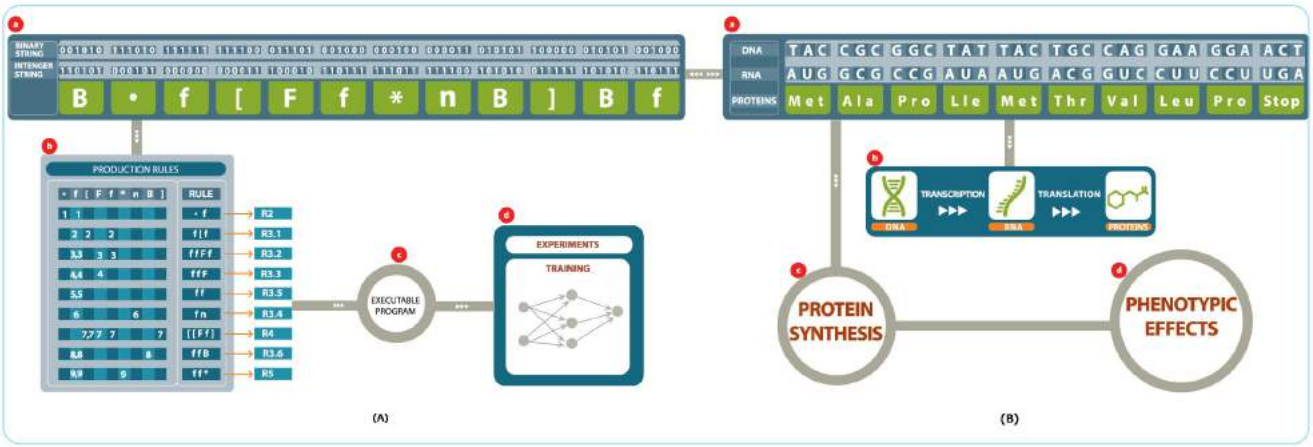


Figure 5: (b) DNA transcription into RNA and translation of RNA into protein. (a) In the analogous artificial process, a binary string is transcribed into an integer string and the string is translated into the production rules of the L-system.

In biological genetic processing Figure 5(b), DNA is transcribed into ribonucleic acid (RNA) and the RNA is translated into proteins. The proteins are derived from linear sequences of amino acids encoded by codons (groups of three nucleotides selected among U, G, A, and G of the genetic code (Table 2). In Figure 5(b), the protein is formed by a sequence of amino acids starting with methionine (Met) and ending with proline (Pro). Such protein synthesis triggers all stages of the neuronal development (phenotypic effects), as shown in Figure 5(b). The elements of the alphabet $\Sigma = \{., f, F, n, [,], *, B, \}$ of the L-system, described in subsection 4.1 and displayed in bold font in Table 2, are a metaphor of the genetic code. Each two-bit sequence represents one nucleotide, for example, the set (00, 01, 10, 11) symbolizes (U, C, A, G) in the original genetic code. Accordingly, six bits represent three nucleotides, that is (000000, 011111) symbolizes (UUU, CGG). The Function-Rule-Extraction-with-GA, translates a string to valid production rules of the L-System (Table 1). Figure 5(a) illustrates, an example of rule extraction for a single individual of the population. In this figure, the transcription string yields the string B.f[Ff.nB]Bf, we seek inside that the shortest string containing all valid rules, in this case, **.f[Ff *nB]**.

4.3 Fitness Assessment

The performance of an ANN can be appropriately measured by the mean square error (MSE), which measures the expected squared distance between the predicted and true values. As such, it measures the quality of a predictor. The system adjusts the weights of its neural networks to minimize the MSE in the training and test sets. ADEANN-Deep aims not only at minimizing the MSE of the ANN in the training set but also properly predict in the test set. The fitness function equation 11 automatically implements an approach that rewards economical ANNs with better prediction capacities and ease of implementation. This function selects networks according to two criteria: number of hidden-layers neurons and MSE. Therefore, smaller networks score higher fitness values than larger networks of similar performance. NEAS evolving LSTM networks are rarely reported in the literature. Our ADEANN-Deep system partially fills this gap and provides accurate, automatic direct and recurrent ANNs for

	00 (U)	01 (C)	10 (G)	11 (A)	
00 (U)	f (UUU)	F (UCU)	n (UAU)	. (UGU)	00 (U)
00 (U)	n (UUC)	. (UCC)	f (UAC)	F (UGC)	01 (C)
00 (U)	F (UUA)	f (UCA)	B (UAA)	f (UGA)	10 (A)
00 (U)	[(UUG)	n (UCG)	[(UAG)	* (UGG)	11 (G)
01 (C)	f (CUU)	] (CCU)	n (CAU)	* (CGU)	00 (U)
01 (C)	* (CUC)	F (CCC)	f (CAC)	F (CGC)	01 (C)
01 (C)	] (CUA)	f (CCA)	* (CAA)	[(CGA)	10 (A)
01 (C)	f (CUG)	* (CCG)	B (CAG)	] (CGG)	11 (G)
10 (A)	* (AUU)	] (ACU)	n (AAU)	f (AGU)	00 (U)
10 (A)	f (AUC))	B (ACC)	f (AAC)	B (AGC)	01 (C)
10 (A)	F (AUA)	[(ACA)	B (AAA)	n (AGA)	10 (A)
10 (A)	* (AUG)	f (ACG)	* (AAG)	] (AGG)	11 (G)
11 (G)	] (GUU)	[(GCU)	F (GAU)	n (GGU)	00 (U)
11 (G)	n (GUC)	B (GCC)	[(GAC)	. (GGC)	01 (C)
11 (G)	f (GUA)	] (CGA)	B (GAA)	F (GGA)	10 (A)
11 (G)	B (GUG)	f (GCG)	* (GAG)	[(GGG)	11 (G)

Table 2: The genetic code from the perspective of mRNA, translated as in Figure 5(b). In the same table, the DNA's metaphor

pattern recognition problems and dynamical systems simulations.

$$fitness = [\exp(-MSE) \times \exp(-NNHL) + \frac{1}{(MSE \times NNHL)}] \tag{11}$$

where MSE is the mean squared error and NNHL is the number of neurons in the hidden layer.

Function-Rule-Extraction-with-GA(String B,String S)

1:Inputs:
2: vars: B:=[IxG] //I is the number of individuals in the population
3:S:=[IxG] //G is the number of desired genes
4:j, M ← B.length();
5:i, N ← S.length();
6:for(i←0, j←0; i<N and j<M; i←i+1)
7: if (S.charAt(i) == B.charAt(j)) j++;
8: else
9: i← -j,j←0
10: if (j = M) return i - M;
11: else
12: return N;

---

5 MATERIAL AND METHODS

5.1 EXPLANATORY VARIABLE SELECTION IN PREDICTIONS MODELS

In prediction models, the explanatory variable can explain or cause diferences in a response variable. After the identification of the explanatory variables, an explanatory variable selection method is applied to find the optimal set of input variables required to describe the behavior of the energy price, which should contain a minimum degree of redundancy. The aim is to test how two or more variables act togheter to affect the output variable and determine whether they improve the prediction of the desired value. The PLDs prediction has the following referenced variables: stored energy in reservoirs (%MLT), inflow energy in reservoirs (%MLT), total hydro generation (MWmed), total thermal generation (MWmed), system power load (MWmed), where %MLT is the long-term average (historical average of 79 years). Below, we detail the variables chosen in the work of [9] for each submarket : North: stored energy, inflow energy and load, Northeast: stored energy, inflow energy, thermal generation and load. Center-Wets/Southeast: stored energy, hydro generation, thermal generation and load, South: stored energy, hydro generation, thermal generation and load. We applied the same type of selection used in the each submarket variables.

5.2 DATASET DESCRIPTION

The dataset used in this research contains the electricity prices data taken from [19] presented on a weekly basis, in addition to the explanatory variables data taken from [20]. In the simulations we applied the dataset constructed by [9] (period from 2002 to 2009) to each submarket:North, Northeast, South and Center-West/Southeast.

5.3 USED METRICS

The hybrid system proposed in this system is applied to the Brazilian electricity market. Some metrics commonly used to evaluate price forecasting accuracy are employed in this paper , Root mean squared error (RMSE) , Mean absolute error (MAE) and Mean absolute percentage error (MAPE), these quantities are calculate by:

$$MAE = \frac{1}{N}\Sigma_{i=1}^{N}|p_i true - p_i forecast| \tag{12}$$

$$RMSE = \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}\Big(p_i true - p_i forecast\Big)^2} \tag{13}$$

$$MAPE = \frac{100\%}{N}\sum_{I=1}^{N}\left|\frac{p_i true - p_i forecast}{p_i true}\right| \tag{14}$$

where: N is the number of samples, $p_i true$ is the actual price and $p_i forecast$ is the forecasted price.

5.4 STATISTICS

We repeated each experiment five times, the independence of the events was assured since the runs were independent and had randomly generated initial seeds. Furthermore, to evaluate the significance of the results obtained from ADEANN and the other NEAs, we carried out t-tests with a confidence level of 95% (i.e., a p-value under 0.05). To statistically compare the performances of two NEAs for the classification problems, we considered the three following criteria: RMSE as the primary criterion, MAPE as the second one, and MAE as the third one. Similarly to [21], without the occurrence of any significant statistical differences between the RMSE, MAPE, and MAE values of two NEAs on a given dataset, it was considered that both algorithms performed equally well. In this case, both algorithms receive 1 point. In contrast, if two algorithms obtain significantly different RMSE or MAPE or MAE scores, the better performing algorithm receive two points and the other zero points. Consequently, H_0 and accept the alternative hypothesis H_1. In deciding whether two performances differ, we test the significance of the difference between u_1 and u_2 ($p < 0.05$). The overall performance of each NEA is then calculated by summing all points achieved in the pairwise comparisons.

5.5 DATA PREPARATION

Many problems are involved in the analysis of rare patterns of ocurrences. As an example, Figure 6(a) shows the histogram of the PLD series for the North region. It shows that some patterns occur more often than others. In addition, most of the time the price remains at low values, under R$100.00, and the energy price rarely reaches values above R$300.00. However, neural networks are sensitive to imbalanced data sets since it causes difficulties in the learning process and can deteriorate the model performance. The data balancing was then applied in this paper during data preparation process. Figures 6 (a) and 6 (b) show the histogram of the PLD series before and after data balancing for the North region.

Therefore, we have tried to replicate the data balancing process adopted in the work of [9]. The datasets reveal that most of the PLD standards have values below R$100.00. Thus, values that could cause noise during training of the neural network, such as outliers or with a low number of examples, were excluded from the datasets. Some projections of the balanced and unbalanced data for the Northeast region are presented below.

The dataset for Center West/Southeast region has 398 records, out of which: 344 tuples with PLD in the interval [0,100], 37 tuples with PLD in the interval [101, 200], 17 tupless with PLD in the interval [201,300]. The technique for removing outliers, called Tukey's method [22], better known as boxplot, consists of sorting the database in ascending order according to the PLD value. Subsequently, it finds the median, the first and the third quartile. With the values of the quartiles found, the interquartile interval is calculated, which consists of the difference between the third and the first quartile, given by equation 15. With the interquartile range found, the method defines lower (equation 16) and upper (equation 17) limits from the interquartile (IQR) and the first and third quartiles. Data outside these limits will be determined as an outlier. After pre-processing, the dataset resulted in 344: 334 tuples with PLD in the interval [0,100], 7 tuples with PLD in the interval [101,200] and 3 tuples with PLD in the interval [201,300].

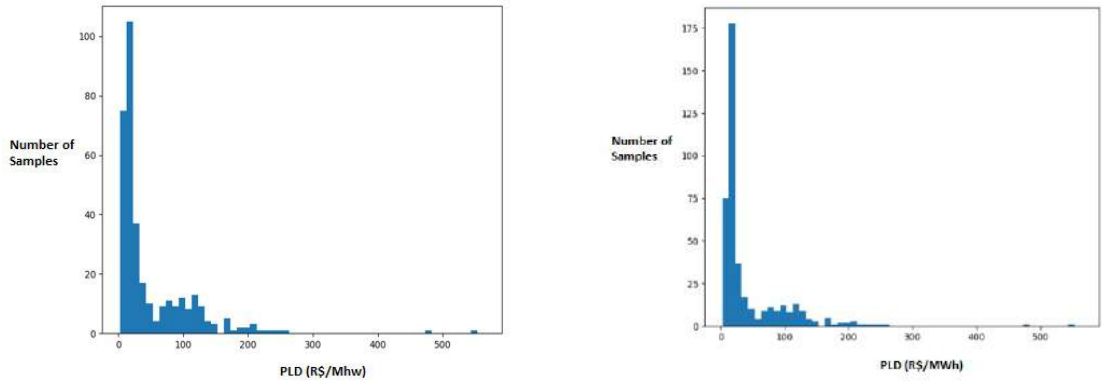
$$IIQ = Q3 - Q1 \tag{15}$$

$$Ouliers < Q1 - 1.5 * IIQ \tag{16}$$

$$Ouliers > Q3 + 1.5 * IIQ \tag{17}$$

6 SIMULATIONS RESULTS

The methodology proposed in this paper is applied to the Brazilian electricity market, considering the five Brazilian regions North, South, Midwest, Southeast and Northeast. In addition, three levels were considered for PLD (Heavy, Medium and Light). The following sections present the obtained simulation results.



(a) Before Data Balancing (b) After Data Balancing

Figure 6: Histrogram of the PLD series to North Region

Variable	Minimun	Mean	Maximum	Std.Deviation
Stored Energy	20.71	65.30	87.64	16.7
Inflow energy	47.86	103.87	182.00	24.38
Hydro generation	8854.14	17.635.02	23.378.14	3253.84
Thermal generation	192.86	1112.57	3258.86	615.93
Load	19.295.57	28.048.88	34.668.00	3148.61
PLD	4.0	74.65	684.00	117.53

Table 3: Summary statistics of the variables from Center-west/Southeast region. Font: [9]

6.1 MULTI LAYER PERCEPTRON NETWORKS ASSESSMENT FOR ENERGY PRICE FORECASTING

In subsections 6.1.1 and 6.1.2 experimental results for deep feedforward networks are presented. We have used two strategies: simulations with unbalanced and balanced data. In addition we have applied some criteria commonly used to assess price forecasting accuracy, such as RMSE, MAE, and MAPE.

6.1.1 UNBALANCED DATA

Tables 4 illustrates the values of RMSE, MAE, and MAPE obtained from ADEANN-Deep for the South, North, Northeast and Southeast regions. Figure 7(a) shows the short-term price observed and predicted with the proposed hybrid system 36-weeks ahead to Northeast region. The neural network architecture returned by ADEANN-Deep, had the following parameters: Number of inputs = 5, number of neurons in the first hidden layer = 131, number of neurons in the second hidden layer = 137, number of outputs = 1, function of activation of the neurons of the first hidden layer = RELU, function of activation of the neurons of the second hidden layer = RELU, function of activation of the output layer = LINEAR, optimizer = ADAM and number of epochs = 10000.

For the Southeast region, the RMSE and MAE obtained from ADEANN-Deep 16.12 R$/Mwh and 7.69 R$/ Mwh were higher than those obtained from the Hybrid System 7.5 R$/Mwh and 5 R$/Mwh [9]. However, the value of MAPE 0.20 R$/Mwh reached using ADEANN-Deep was below that obtained from the Hybrid System [9], which generated a MAPE value equal to 4 R$/ Mwh. For the Northeast region, the RMSE and MAE achieved using ADEANN-Deep 9.41 R$/Mwh and 6.66 R$/ Mwh were higher than those from the Hybrid System 8.25 R$/Mwh and 3.75 R$/Mwh [9]. However, the value of MAPE 0.19 R$/Mwh generated from ADEANN-Deep was below that from the Hybrid System [9], which reached a MAPE value equal to 4.4 R$/ Mwh.

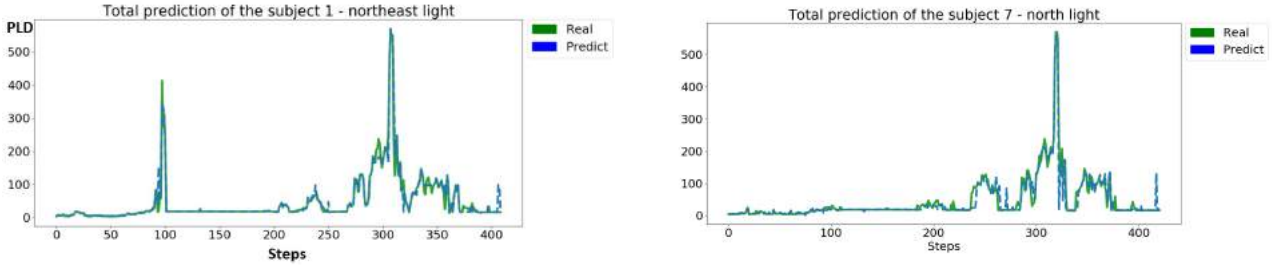
Figure 7(b) shows the short-term price observed and predicted with the proposed hybrid system 36-weeks ahead to North region. For the South region, the RMSE and MAE generated from ADEANN-Deep 11.35 R$/Mwh and 8.02 R$/ Mwh were above those from the Hybrid System 9 R$/Mwh and 3.75 R$/Mwh [9]. However, the value of MAPE 0.21 R$/Mwh reached using ADEANN-Deep was lower than that achieved using the Hybrid System [9], which generated a MAPE value equal to 5 R$/ Mwh. For the North region, the RMSE and MAE obtained from ADEANN-Deep 9.70 R$/Mwh and 6.86 R$/ Mwh were higher than those from the Hybrid System 7.5 R$/Mwh and 3 R$/Mwh [9]. However, the value of MAPE 0.20 R$/Mwh generated using ADEANN-Deep was below that obtained from the Hybrid System [9], which reached a MAPE value equal to 4.75 R$/ Mwh.

6.1.2 BALANCED DATA

Table 5 illustrates the values of RMSE, MAE, and MAPE obtained from ADEANN-Deep for the South, North, Northeast and Southeast regions. The analysis of the results for the South region revealed a mean square error (RMSE) generated using ADEANN-Deep of 8.20 R$ Mwh, lower than 9 R$/Mwh, obtained from the Hybrid model [9]. The MAE and MAPE generated

	South Region			North Region		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Medium PLD	11.40	8.05	0.21	9.35	6.61	0.22
Light PLD	11.07	7.82	0.20	9.27	6.55	0.20
Heavy PLD	11.59	8.20	0.21	10.48	7.41	0.22
Average	11.35	8.02	0.21	9.7	6.86	0.22
	Northeast Region			Southeast Region		
Medium PLD	9.03	6.38	0.19	15.47	10.94	0.20
Light PLD	10.21	7.22	0.21	15.47	10.94	0.20
Heavy PLD	9.01	6.37	0.18	17.15	12.13	0.22
Average	9.41	6.66	0.19	16.12	7.69	0.20

Table 4: Energy price error measures obtained with the proposed hybrid system 36-weeks ahead - (Unbalanced Data)



(a) Northeast Region (b) North Region

Figure 7: Energy price observed and predicted with ADEANN-DEPP (Light PLD) - Unbalanced Data

using ADEANN-Deep were 5.80 R$/ Mwh and 0.19 R$/Mwh, respectively, while the values obtained from the hybrid method [9] were 3.75 R$/ Mwh and 5 R$/ Mwh, respectively. Therefore, our MAE value was higher than that obtained from the hybrid system, while our MAPE value was below it. For the North region, the mean square error (RMSE) generated from ADEANN-Deep was 5.02 R$/Mwh, lower than 7.5 R$/Mwh, reached using the hybrid system [9]. The MAE and MAPE obtained using ADEANN-Deep were 3.55 R$/ Mwh and 0.12 R$/ Mwh, respectively, while the values generated using the hybrid model [9] were 3 R$/Mwh and 4.75 R$/Mwh, respectively. Therefore, our MAE value was higher than that from the hybrid system and our MAPE value was below it. The analysis of the results for the Southeast region revealed a mean square error (RMSE) using the ADEANN-Deep of 9.8 R$ Mwh, higher than 7.5 R$/Mwh, obtained from the Hybrid model [9]. The MAE and MAPE reached using ADEANN-Deep were 6.93 R$/ Mwh and 0.14 R$/Mwh, respectively, while the values from the hybrid method [3] were 5 R$/ Mwh and 4 R$/ Mwh, respectively. Therefore, our MAE value was higher than that obtained from the hybrid system and our MAPE value was below it. For the Northeast region, the mean square error (RMSE) generated from ADEANN-Deep was 5.8 R$/Mwh, lower than 8.75 R$/Mwh, obtained from the hybrid model [9]. The MAE and MAPE generated using ADEANN-Deep were 4.1 R$/ Mwh and 0.17 R$/ Mwh, respectively, while the values obtained from the hybrid model [9] were 3.75 R$/Mwh and 4.4 R$/Mwh, respectively. Therefore, our MAE value was higher than that reached using the hybrid system and our MAPE value was below it. It is noteworthy that all other models are forecasting 12 weeks ahead, however, ADEANN-Deep is forecasting 36 weeks ahead. Figure 8 shows the short-term price observed and predicted with the proposed hybrid system 36-weeks ahead to North region.

	South Region			North Region		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Medium PLD	7.52	5.31	0.18	4.22	2.98	0.11
Light PLD	8.52	6.02	0.20	5.2	3.68	0.13
Heavy PLD	8.57	6.06	0.19	5.64	3.99	0.14
Average	8.20	5.80	0.19	5.02	3.55	0.12
	Northeast Region			Southeast Region		
Medium PLD	5.86	4.14	0.17	9.91	7.01	0.15
Light PLD	5.70	4.03	0.16	9.95	7.03	0.15
Heavy PLD	5.84	4.13	0.17	9.55	6.75	0.14
Average	5.80	4.10	0.17	9.8	6.93	0.14

Table 5: Energy price error measures obtained with the proposed hybrid system 36-weeks ahead - (Balanced Data)

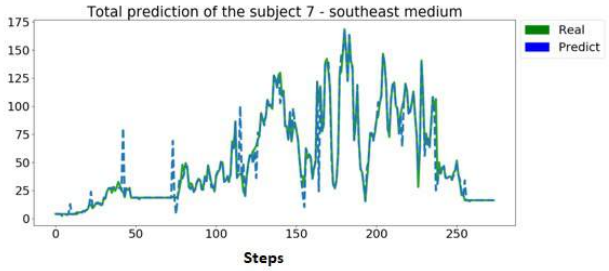
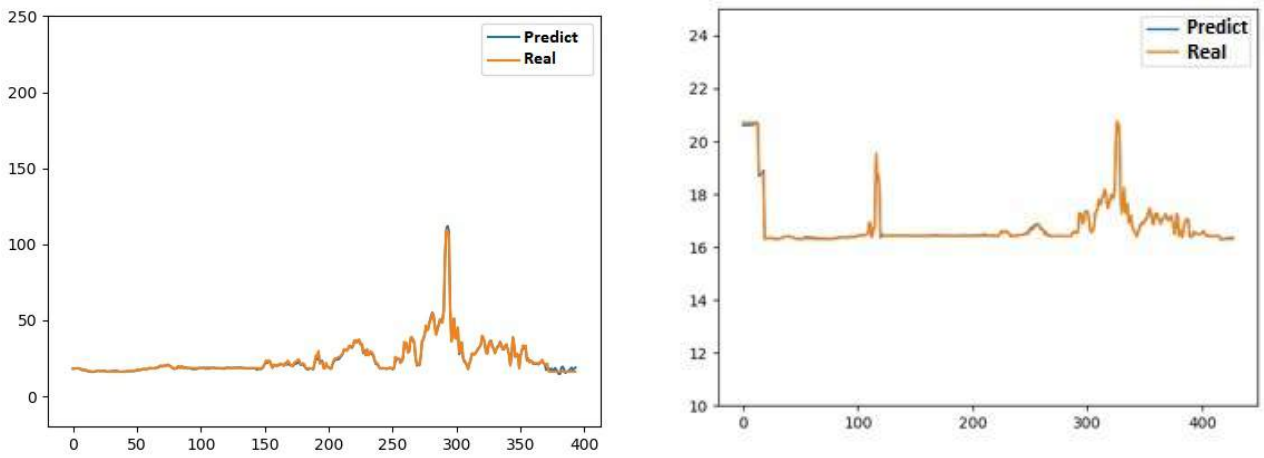


Figure 8: Energy price observed and predicted with the hybrid model to Southeast Region (heavy PLD) - Balanced Data

6.2 LSTM NETWORKS ASSESSMENT FOR ENERGY PRICE FORECASTING

Recurrent neural networks have gained widespread use in modeling sequential data. The Long Short-Term Memory network (LSTM network) is a type of RNN in deep learning because very large architectures can be successfully trained. The work [23] shows the robustness of LSTM networks in handling unbalanced data. The empirical studies conducted and reported in the paper [24] show that deep learning based algorithms such as LSTM outperform traditional-based algorithms like ARIMA model. After simulations with feedforward neural networks, presented in sections 6.1.1 and 6.1.2, using unbalanced and balanced data, respectively, we started simulations with LSTM networks, which are more robust for time series prediction. In order to verify the robustness of LSTM networks, in our simulations we haven't used data balancing.



(a) Southeast region

(b) Northeast region

Figure 9: Energy price observed and predicted with ADEANN-DEPP - Unbalanced Data - LSTM networks

Table 6 illustrates the values of RMSE, MAE and MAPE obtained from ADEANN-Deep for the Northeast and Southeast regions. The analysis of the results for the Northeast region, considering only heavy PLD, revealed a mean square error RMSE, MAE and MAPE using the ADEANN-Deep of 0.21 R$ Mwh, 0.047 R$/ Mwh and 0.20 R$/ Mwh , respectively, lower than the values from simulations with multilayer perceptron networks (MLPs), see Table 4, 9.01 R$/ Mwh, 6.37 R$/ Mwh and 0.18 R$/ Mwh, respectively. The neural network architecture returned by ADEANN-Deep had the following parameters: Number of inputs = 5, number of neurons in the first hidden layer = 102, number of neurons in the second hidden layer = 87, number of outputs = 1, function of activation of the neurons of the first hidden layer = hyperbolic tangent, function of activation of the neurons of the second hidden layer = hyperbolic tangent, function of activation of the output layer = LINEAR, optimizer = Adam, Batch=32, and number of epochs = 2000.

The analysis of the results for the Northeast region, considering all the PLDs (Heavy, Medium and Light), revealed a RMSE, MAE and MAPE using the ADEANN-Deep lower than the values from simulations with Hybrid Systems [9], considering the average among the PLDs (Heavy, Medium and Light), whose values were 8.25 R$/Mwh, 3.75 R$/Mwh and 4.4 R$/ Mwh, respectively.

The analysis of the results for the Southeast region, considering only heavy PLD, revealed a mean square error RMSE, MAE and MAPE using the ADEANN-Deep of 0.27 R$ Mwh, 0.07 R$/ Mwh and 0.39 R$/ Mwh , respectively, lower than the values from simulations with multilayer perceptron networks (MLPs), see Table 4, 17.15 R$/ Mwh, 12.13 R$/ Mwh and 0.22 R$/ Mwh, respectively. Figure 9(a) and Figure 9(b) show the short-term price observed and predicted with the proposed hybrid system 36-weeks ahead to Southeast and Northeast regions using LSTM network.

The analysis of the results for the Southeast region, considering all the PLDs (Heavy, Medium and Light), revealed a RMSE, MAE and MAPE using the ADEANN-Deep lower than the values, from simulations with Hybrid Systems [9], considering the

	Southeast Region			Northeast Region		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
12 weeks ahead/Heavy PLD	0.07	0.15	0.26	0.023	0.007	0.15
24 weeks ahead/Heavy PLD	0.14	0.76	0.37	0.035	0.17	0.19
36 weeks ahead//Heavy PLD	0.07	0.39	0.27	0.047	0.20	0.21
12 weeks ahead/Medium PLD	0.07	0.14	0.26	0.026	0.008	0.16
24 weeks ahead//Medium PLD	0.08	0.46	0.29	0.29	1.44	0.54
12 weeks ahead/light PLD	0.005	0.07	0.14	0.012	0.004	0.11
	North Region			South Region		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
12 weeks ahead/Heavy PLD	0.047	0.07	0.21	0.06	0.13	0.25
24 weeks ahead/Heavy PLD	0.035	0.10	0.18	0.04	0.25	0.21
36 weeks ahead//Heavy PLD	0.058	0.36	0.24	0.14	0.85	0.38
12 weeks ahead/Medium PLD	0.03	0.044	0.17	0.09	0.20	0.30
24 weeks ahead/Medim PLD	0.05	0.15	0.23	0.07	0.40	0.27
36 weeks ahead/Medim PLD	0.10	0.66	0.32	0.13	0.87	0.35
12 weeks ahead/Light PLD	0.084	0.14	0.29	0.04	0.082	0.21
24 weeks ahead/Light PLD	0.056	0.15	0.23	0.058	0.32	0.24
36 weeks ahead/Light PLD	0.07	0.42	0.27	0.08	0.48	0.28

Table 6: Energy price error measures obtained with the proposed system free simulations for 12, 24 and 36 weeks ahead - (Unbalanced Data)

average among the PLDs (Heavy, Medium and Light), 7.5 R$/Mwh, 5 R$/Mwh and 4 R$/Mwh, respectively.

The results for the North region, considering the average of the PLD values (Heavy, Medium, Light), revealed a mean square error MAE, MAPE and RMSE using the ADEANN-Deep of 0.076 R$ Mwh, 0.48 R$/ Mwh and 0.27 R$/ Mwh , respectively, lower than the values from simulations with multilayer perceptron networks (MLPs), see Table see Table 4 , 6.86 R$/ Mwh, 0.22 R$/ Mwh and 9.7 R$/ Mwh, respectively.

The analysis of the results for the North region, considering all the PLDs (Heavy, Medium and Light), revealed a RMSE, MAE and MAPE using the ADEANN-Deep lower than the values from simulations with Hybrid Systems [9], considering the average among the PLDs (Heavy, Medium and Light), R$/ 7.5 Mwh, R$/ 3 Mwh and R$/ 4.75 Mwh, respectively.

The results for the South region, considering the average of the PLD values (Heavy, Medium, Light), revealed a mean square error MAE, MAPE and RMSE using the ADEANN-Deep of 0.12 R$/Mwh, 0.73 R$/ Mwh and 0.34 R$/ Mwh, respectively, lower than the values from simulations with multilayer perceptron networks (MLPs), see Table 4, 8.02 R$/ Mwh, 0.21 R$/ Mwh and 11.35 R$/ Mwh, respectively.

The analysis of the results for the South region, considering all the PLDs (Heavy, Medium and Light), revealed a RMSE, MAE and MAPE using the ADEANN-Deep lower than the values, from simulations with Hybrid Systems [9], considering the average among the PLDs (Heavy, Medium and Light), 9 R$/ Mwh, 3.75 R$/Mwh and 5 R$/Mwh, respectively.

The results using HIRA model [14] in the Hungarian market, three days in advance for day d, at the 3rd interval for base load energy price (is the average hourly price for all 24 h) and peak (is the delivery of the same amount of energy over the period from 08:00 to 20:00) indicate a percentage error around 0.56% to 2.96%. The percentage hourly price forecast error is around 4.55% to 11.37%. Our results for the all regions revealed a percentage error around 0.07% to 1.44%, lower than the values from HIRA model [14].

The hyperparameters such as learning rate, activation functions, epochs, batch, and so on, aren't part of the individuals. Hyperparameters are chosen randomly, after mapping the genotype to the phenotype, Table 7 illustrates the Hyperparameters that are assigned to some individuals in one of the experiments carried out for the Northeast region. Some values of hyperparameters used in the simulations were: optimizer (rmsprop-Adam), NINT1 in the range [80,120], NINT2 in the range [90,120], learning rate in the range [0.06,0.1], activation function (tanh,Linear,ReLU). ADEANN-Deep selects the best individuals of each generation and classifies them in descending order of fitness, at the end it selects the best individual of all. In the simulation for the Northeast region, the best ranked individual is an LSTM network with the hyperameters shown in row one of table 7. That is, notice that ADEANN-Deep selected this network, since it has a total of 189 neurons in the two intermediate layers and an MSE value of 0.0009 (ie in the order of 10E-4). Note that there is another smaller network in line 7 of table 7, with 180 neurons in the two intermediate layers, but this network obtained an MSE value of 0.005 (that is, in the order of 10E-3). Therefore, despite smaller, this network has less predictive capacity than the first one. Thus, the fitness function ranks and selects individuals by establishing a balance between the two hyperparameters (number of neurons in the intermediate layer and MSE).

7 CONCLUSIONS

In this paper a hybrid approach is proposed for a short-term energy price prediction. The model considers multi-step 36 week-ahead price prediction applied to the Brazilian electricity market. The results obtained with the use of deep feedforward

Epochs	Batch	Optimizer	Ac.Func.InterI (G)	Ac.Func.InterII	Ac.Func.Output	Learning Rate	NINT1	NINT 2	MSE	Fitness
2000	32	Adam	tanh	tanh	Linear	0.08	102	87	0.0009	5.60
2000	32	rmsprop	tanh	tanh	Linear	0.1	100	100	0.0018	2.77
10000	32	Adam	tanh	tanh	Linear	0.1	102	107	0.0022	2.12
10000	32	rmsprop	tanh	tanh	Linear	0.1	107	114	0.003767	1.20
10000	32	rmsprop	tanh	tanh	Linear	0.1	105	120	0.003871	1.18
2000	32	rmsprop	tanh	tanh	Linear	0.06	101	100	0.00519	0.957
10000	32	rmsprop	tanh	tanh	Linear	0.1	80	100	0.00594	0.935
10000	32	Adam	tanh	tanh	Linear	0.1	100	115	0.00735	0.632
2000	32	rmsprop	tanh	tanh	Linear	0.95	103	91	0.0088	0.586
10000	32	rmsprop	tanh	tanh	Linear	0.1	105	120	0.00854	0.520

Table 7: Hyperparameters assigned to individuals, where: Ac.Func.InterI is the activation function of the first intermediate layer, Ac.Func.InterII is the activation function of the second intermediate layer, Ac.Func.Output : is the activation function of output layer, NINT1 an d NINT2 are the numbers of neurons of the first and second intermediate layers

networks are compared with the study of [9] and other methods in sections 6.1.1 and 6.1.2. The results obtained from ADEANN-Deep applied to the Brazilian market presented a sufficiently good accuracy level compared to other methods. Data balancing and the use of explanatory variables were essential for having improved the results generated using ADEANN-Deep, according to the results presented in section 6.1.2. Statistical test (t-test) with confidence level of 95% shows that in 58.33% of the cases ADEANN-Deep provides better results than the hybrid system [9].

It is common to find in the literature that LSTM networks are more robust than deep feedforward networks, a hypothesis confirmed in our results, but the main findings of our work relate to the use of a hybrid neural system, which automatically selects efficient LSTM networks that are able to properly predict PLD for the Brazilian market. Thus, the use of LSTM networks has expanded the possibilities of solving the problem using recurrent neural networks. In addition, the results obtained by our system were compared with those of another hybrid system [9], which made predictions in the Brazilian market based on the results and discussions presented in section 2.3.2, it is concluded that our system is more accurate.

In addition, we compared the average percentage error obtained by our system with the HIRA method [14], which was also applied and tested on two eletricity markets: German and Hungarian. It can be concluded from the results presented in section 2.3.2 that the LSTM Network can predict the value of the PLD with an error of up to 1.44%, which is lower than the value generated from multilayer perceptron networks and HIRA model [14].

In further works, we will tackle other challenges, such as a way to train neural networks in a GPU architecture will be needed, which will provide a significant computation speed-up over CPU-only training. Furthermore, it is important to implement other approaches to improve the performance of deep neural network learning algorithms, such as the Ensemble Learning Methods for Deep Learning Neural Networks.

Acknowledgment

We would like to thank the Federal University of Pará for having supported this study.

REFERENCES

[1] A. Tealab, H. Hefny and A. Badr. “Forecasting of nonlinear time series using ANN”. *Future Computing and Informatics Journal*, vol. 2, no. 1, pp. 39 – 47, 2017.

[2] J. M. P. Menezes and G. A. Barreto. “Long-term time series prediction with the NARX network: An empirical evaluation”. *Neurocomputing*, vol. 71, no. 16, pp. 3335 – 3343, 2008. Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006).

[3] P. Li, K. Zhou, X. Lu and S. Yang. “A hybrid deep learning model for short-term PV power forecasting”. *Applied Energy*, vol. 259, pp. 114216, 2020.

[4] Z. Peng, S. Peng, L. Fu, B. Lu, J. Tang, K. Wang and W. Li. “A novel deep learning ensemble model with data denoising for short-term wind speed forecasting”. *Energy Conversion and Management*, vol. 207, pp. 112524, 2020.

[5] R. F. Alexandre, F. Campelo and J. ao A. Vasconcelos. “Multi-objective evolutionary algorithms for the truck dispatch problem in open-pit mining operations”. *Learning & Nonlinear Models*, vol. 17, no. 2, pp. 53–66, 2019.

[6] G. Hornby and J. B. Pollack. “Creating High-Level Components with a Generative Representation for Body-Brain Evolution.” *Artif. Life*, vol. 8, no. 3, pp. 223–246, 2002.

[7] D.-W. Lee, S.-W. Seo and K.-B. Sim. “Evolvable Neural Networks Based on Developmental Models for Mobile Robot Navigation.” *Int. J. Fuzzy Logic and Intelligent Systems*, vol. 7, no. 3, pp. 176–181, 2007.

[8] L. M. L. de Campos, R. C. L. de Oliveira and G. A. L. de Campos. “A Neurogenetic Algorithm Based on Rational Agents.” *Computing and Informatics*, vol. 37, no. 5, pp. 1073–1102, 2018.

[9] J. C. R. Filho, C. de M. Affonso and R. C. de Oliveira. "Energy price prediction multi-step ahead using hybrid model in the Brazilian market". *Electric Power Systems Research*, vol. 117, pp. 115 – 122, 2014.

[10] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy and B. Hodjat. "Chapter 15 - Evolving Deep Neural Networks". In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, edited by R. Kozma, C. Alippi, Y. Choe and F. C. Morabito, pp. 293 – 312. Academic Press, 2019.

[11] S. S. Tirumala, S. Ali and C. P. Ramesh. "Evolving deep neural networks: A new prospect." In *ICNC-FSKD*, pp. 69–74. IEEE, 2016.

[12] M. Gutoski, L. T. Hattori, N. M. R. Aquino, H. C. M. Senefonte, M. Ribeiro, A. E. Lazzaretti and H. S. Lopes. "Quantitative Analysis of Deep Learning Frameworks". *Learning & Nonlinear Models*, vol. 15, no. 1, pp. 45–52, 2017.

[13] M. Cai, M. Pipattanasomporn and S. Rahman. "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques". *Applied Energy*, vol. 236, pp. 1078 – 1088, 2019.

[14] D. M. Cerjan M., Petricic A. "Hira Model for Short-Term Electricity Price Forecasting". *Energies*, vol. 12, no. 8, pp. 568–600, 2019.

[15] J. Junior and J. Machado. *The Energy Regulation and Markets Review*. Law Business Research Ltd, Eighth Edition, London, 2019.

[16] T. Community. *https://tex.stackexchange.com/questions/432312/how-do-i-draw-an-lstm-cell-in-tikz/432344*, 2020 (accessed February 18 2020).

[17] A. Élida Nogueira Frauches Almoaia, W. F. Sacco and A. J. Silva Neto. "Hybrid differential evolution with the topographical heuristic". *Learning & Nonlinear Models*, vol. 17, no. 2, pp. 42–52, 2019.

[18] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[19] *Website of Brazilian Electrical Energy Commercialization Chamber*, 2020 (accessed February 14 2020).

[20] *Website of Brazilain National System Operator*, 2020 (accessed February 14 2020).

[21] M. Castellani. "Evolutionary generation of neural network classifiers - An empirical comparison." *Neurocomputing*, vol. 99, pp. 214–229, 2013.

[22] A. Dagdas. "Heat exchanger optimization for geothermal district heating systems: A fuel saving approach". *Renewable Energy*, vol. 32, no. 6, pp. 1020 – 1032, 2007.

[23] S. Akkaradamrongrat, P. Kachamas and S. Sinthupinyo. "Text Generation for Imbalanced Text Classification". In *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 181–186, July 2019.

[24] S. Siami-Namini, N. Tavakoli and A. S. Namin. "A Comparison of ARIMA and LSTM in Forecasting Time Series." In *ICMLA*, edited by M. A. Wani, M. M. Kantardzic, M. S. Mouchaweh, J. Gama and E. Lughofer, pp. 1394–1401. IEEE, 2018.