Comparing Deep and Dendrite Neural Networks: A Case Study

Gerardo Hernández[1], Erik Zamora[2], and Humberto Sossa[1(✉)]

[1] Instituto Politécnico Nacional - CIC,
Av. Juan de Dios Batiz S/N, Gustavo A. Madero, 07738 Mexico City, Mexico
ghernandez_a13@sagitario.cic.ipn.mx, hsossa@cic.ipn.mx
[2] Instituto Politécnico Nacional - UPIITA,
Av. Instituto Politécnico Nacional 2580, Barrio la Laguna Ticoman,
Gustavo A. Madero, 07340 Mexico City, Mexico
ezamorag@ipn.mx

Abstract. In this paper, a comparative study between two different neural network models is performed for a very simple type of classificaction problem in 2D. The first model is a deep neural network and the second is a dendrite morphological neuron. The metrics to be compared are: training time, classification accuracies and number of learning parameters. We also compare the decision boundaries generated by both models. The experiments show that the dendrite morphological neurons surpass the deep neural networks by a wide margin in terms of higher accuracies and a lesser number of parameters. From this, we raise the hypothesis that deep learning networks can be improved adding morphological neurons.

1 Introduction

In the area of Artificial Intelligence there is a great diversity of algorithms for pattern classification, and one of the most important is the Multi-Layer Perceptron (MLP) which through a training process adjusts the hyperplanes of each neuron in each layer to separate the classes of some dataset [22,25,26]. The training is often based on gradient descent and back-propagation [22]. This model since its appearance in 1961 [25] has been widely used in the area of pattern recognition. However, there are other classification algorithms such as Dendrite Morphological Neuron (DMN) which use a training algorithm completely different from back-propagation [22], in the sense that they do not try to approximate a hyperplane through an iterative training process, analyzing each sample of the training set. Instead, this type of neuron analyzes the elements as a complete set and based on lattice operations generate hyperboxes. They are able to classify the different classes of the training set with a higher rate.

The success of Deep Neural Networks (DNN) is well known for recognizing objects in images [12] and speech in audio [9]. The mathematical operations employed in these neurons remain the same as those of a MLP [22]: sums, multiplications and some well-known non-linear functions. Furthermore, convolutions

J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2017, LNCS 10267, pp. 32–41, 2017.
DOI: 10.1007/978-3-319-59226-8_4

are used for reducing the number of learning parameters [13]. So, the novelty of the last 10 years has focused on more computing power, more layers, more data and the dropout [14,23]. These last two are to avoid overfitting in deep models that previously prevented the MLPs from giving better results than the Support Vector Machines (SVM) [3]. It is important to note that these developments are not related to the mathematical structure. This leads us to ask if there are other mathematical operations that can improve the recognition performance. In this paper, we started a research project in that direction. In particular, we compared DNNs with DMNs for a specific type of problem: multi-class spirals with several loops in 2D. Even when this classification problem is artificial, it is useful for studying the essential properties of the two models. A very first analysis was published in [27]; here we extend the analysis for deeper models and more classes. As classification tools, both models are subjects for comparison in terms of percentage of classification and training times, which depend directly on the number of parameters that constitute the model.

The rest of the paper is organized as follows. Section 2 provides a brief description of works that have proposed a different mathematical structure from the mainstream of neural networks. Sections 3 and 4 present the architecture of DNNs and DMNs, respectively. Section 5 discusses the experimental results. Then, in Sect. 6 we give our conclusions and future work.

2 Previous Work

Currently, there are few studies aimed at improving the mathematical structure of deep neural networks. However, before the term "deep learning" was born, we could find several papers with interesting proposals. Pessoa and Maragos [18] combined linear with rank filters. This architecture has shown that it can recognize digits in images, generating similar or better results compared to classical MLPs in shorter training times. Ivakhnenko [11] proposes a multilayer of polynomials to approximate the decision boundary for clasification problems. This was the first deep learning model published in literature. Dubin and Rumelhart [4] introduce product units into neural networks. These units add complexity to the model in order to use less layers. Other mathematical structures have been proposed such as: higher-order neural networks (NNs) [5], sigma-pi NNs [8], second-order NNs [17], functionally expanded NNs [10], wavelet NNs [29] and Bayesian NNs [16]. Glorot investigated more effective ways of training very deep neural networks using ReLUs as activation functions, achieving results comparable to the state-of-the-art [6]. Bengio [2] argues that in order to learn complex functions through training by gradient descent, it is necessary to use deep architectures. In [1] Bengio also analyzes and considers alternatives to training by standard gradient descent, due to the trade-off between efficient learning and latching on information. In this paper, we evaluate the performance of the DNNs with that of the DMNs to show some limitations of the DNNs and how morphological operations could improve deep learning.

3 Deep Neural Networks

"A deep learning architecture is a multilayered stack of simple modules with multiple non-linear layers" [14] (usually between 5 and 20 layers), and each layer contains a n_i number of modules, where i is the layer number, each module is a neuron with some activation function such as sigmoid or tanh. So an MLP and its generalization a DNN are defined by a set of neurons divided into layers: an input, one or more intermediate and an output layer. Thus, the DNN architectures that are constructed to classify the datasets are neural networks which have an i number of intermediate layers and a n_i number of neurons per layer, and the numbers of neurons per layer n_{i-1} and n_i are not necessarily the same. In our experiments we used the Rectified Linear Unit (ReLU) due to better results in DNN according to [6,14,15], so that a neuron is defined by:

$$f(x) = \max\left(0, w^T x\right), \tag{1}$$

where x is the input vector of N dimensions and w is the weights vector that multiplies the input vector. In the output layer, the activation function is changed by a softmax, which is commonly used to predict the probabilities associated with a multinoulli distribution [7], which is defined by

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}, \tag{2}$$

The general DNN architecture is shown in Fig. 1. It is also common practice to vary the number of neurons contained in each layer of the DNN. The training method used for the DNN is Nesterov gradient descent with a mini-batch size of 64 and a moment of 0.9, which helps us to a more stable and fast convergence.

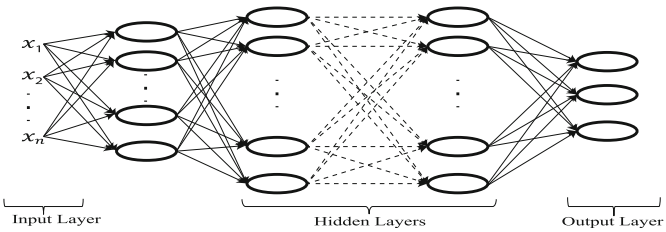

Fig. 1. Architecture of a DNN, where the number of hidden layers is another hyper-parameter.

4 Dendrite Morphological Neurons

A DMN segments the input space into hyperboxes of N dimensions. The output y of a neuron is a scalar given by

$$y = \underset{k}{argmax}\,(d_{n,k}), \tag{3}$$

where n is the dendrite number, k is the class number, and $d_{n,k}$ is the scalar output of a dendrite given by

$$d_{n,k} = \min_i \left(\min \left(x - w^n_{min}, w^n_{max} - x \right) \right), \tag{4}$$

where x is the input vector, w_{min} and w_{max} are dendrite weight vectors. The min operations together check if x is inside the hyperbox limited by w_{min} and w_{max} as the extreme points (see Fig. 2). If $d_{n.k} > 0$, x is inside the hyperbox, If $d_{n,k} = 0$, x is somewhere in the hyperbox boundary; otherwise, it is outside. A good property of DMN is that they can create complex non-linear decision boundaries that separate classes with only one neuron [20,21]. The reader can consult [28] for more information.

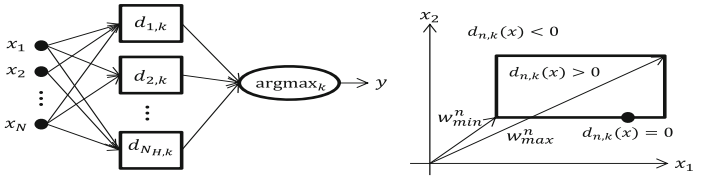

Fig. 2. Dendrite morphological neuron and an example of a hyperbox in 2D generated by its dendrite weights. The hyperbox divides the input space for classification purposes.

The training goal is to determine the number of hyperboxes and their weights needed to classify an input pattern. The regularized divide and conquer training method [28] consists of only two steps. The algorithm begins by opening an initial hyperbox H_0 that encloses all the samples with a margin distance M respect to each side of H_0 to have a better noise tolerance. Next the divide and conquer strategy is executed in a recursive way. The algorithm chooses a training sample x to generate a sub-hyperbox H_{sub} around it. Next it extracts the samples $(X_{H_{sub}}, T_{H_{sub}})$ from (X, T) that are enclosed in H_{sub}, where X is a training samples set represented as a matrix $X \epsilon \Re^{NxQ_{train}}$, Q_{train} is the number of training samples and the target class for each sample is contained in vector $T \epsilon \Re^{1xQ_{train}}$. The recursion divides H_0 until the error rate $E_{\%}$ in the hyperbox H is less or equal to the hyper-parameter E_0. The error rate is defined as $E_{\%} = \frac{|X_{mode}|}{|X|}$, where X_{mode} is the set of the most repeated training class [19]. At the end of the recursion process, the deepest hyperbox is assigned to the ruling class, which is set to the statistical mode of T. The recursive closing procedure is executed by appending all generated sub-hyperboxes with their corresponding classes. The hyperboxes with a common hyperface are joined. A complete description of this training method can be found in [24,28].

5 Experiments

The experiments were designed with the aim of comparing the performance of the two neural networks, taking as a starting point the same training set. The aspects evaluated are the classification accuracy in the validation set, the training time, the number of parameters necessary for the network to correctly classify the training set, and the decision boundaries.

5.1 Spiral Datasets

The training set is a set of synthetic data, designed to test the ability of the two types of neural networks in the unraveling of the hyperplanes, that is, the synthetic data is generated with a high rate of entanglement, and a low degree of overlap between classes. For this purpose the generated data spiral consists of 1 to 5 classes wrapped one over the other, and the number of turns vary between 1 and 10. The representation of said training set is shown in Fig. 3 in such a way that the training set is shaped as shown in Table 1.

5.2 Experimetal Results for DNNs

In order to classify the patterns presented in the Sect. 5.1 the DNN architecture varies in depth the number of neurons per layer, as well as the number of hidden layers, leaving the hyper-parameters fixed to the following values, learning

Table 1. Datasets for spirals with different number of classes $N_C = \{2, 3, 4, 5, 10\}$ and increasing number of loops $N_L = \{1, 2, ..., 10\}$. The number of training patterns is Q_{train} and the number of validation patterns is Q_{val}.

Name	N_C	N_L	Q_{train}	Q_{val}
1.1	2	1	10000	2500
1.2	2	2	10000	2500
1.3	2	3	10000	2500
1.4	2	4	10000	2500
1.5	2	5	10000	2500
1.6	2	6	40000	10000
1.7	2	7	50000	12500
1.8	2	8	60000	15000
1.9	2	9	60000	15000
1.10	2	10	60000	15000

Name	N_C	N_L	Q_{train}	Q_{val}
2.1	3	1	75000	18750
2.2	3	2	75000	18750
2.3	3	3	75000	18750
2.4	3	4	90000	22500
2.5	3	5	90000	22500

Name	N_C	N_L	Q_{train}	Q_{val}
3.1	4	1	120000	30000
3.2	4	2	120000	30000
3.3	4	3	120000	30000
3.4	4	4	120000	30000

Name	N_C	N_L	Q_{train}	Q_{val}
4.1	5	1	150000	37500
4.2	5	2	150000	37500
4.3	5	3	150000	37500
4.4	5	4	150000	37500
4.5	5	5	150000	37500

Name	N_C	N_L	Q_{train}	Q_{val}
5.1	10	1	100000	25000
5.2	10	2	100000	25000
5.3	10	3	100000	25000
5.4	10	4	100000	25000
5.5	10	5	100000	25000

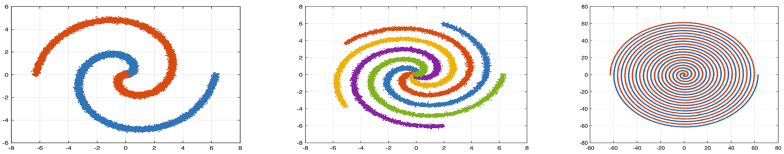

Fig. 3. Spiral of two interlaced spin classes (left), spiral of five classes of one spin per class (center), spiral of two classes with 10 turns each class.

Table 2. Experimental results for DNNs.

Dataset	N_p	T_a	V_a	T_t
1.1	102	0.9947	0.9944	55.68
1.2	402	0.9956	0.9953	47.88
1.3	1332	0.9986	0.998	202.63
1.4	15552	0.998	0.988	335.67
1.5	55952	0.9564	0.9475	590.43
1.6	76152	0.9038	0.8857	3722.79
1.7	211952	0.8302	0.81154	1026.20
1.8	234602	0.776	0.7721	1185.75
1.9	257252	0.7544	0.7306	1508.66
1.10	279902	0.7278	0.7083	2476.12

Dataset	N_p	T_a	V_a	T_t
2.1	252	0.9886	0.9882	35.46
2.2	402	0.9844	0.9915	64.35
2.3	11332	0.9933	0.9745	329.06
2.4	35752	0.9971	0.9917	361.26
2.5	55952	0.8696	0.8416	689.51

Dataset	N_p	T_a	V_a	T_t
3.1	252	0.9839	0.9836	45.46
3.2	502	0.9892	0.9812	205.03
3.3	11432	0.9933	0.9943	201.45
3.4	45852	0.9956	0.9806	437.41
3.5	55952	0.8696	0.8416	565.82

Dataset	N_p	T_a	V_a	T_t
4.1	252	0.9808	0.9794	311.89
4.2	502	0.9831	0.9595	256.71
4.3	25652	0.9934	0.9916	523.81
4.4	35752	0.993	0.9556	749.26
4.5	66052	0.9052	0.8131	314.50

Dataset	N_p	T_a	V_a	T_t
5.1	45852	0.9477	0.9447	78.57
5.2	45852	0.9635	0.9444	91.54
5.3	96352	0.9558	0.9176	455.31
5.4	126652	0.9364	0.8397	598.68
5.5	106542	0.8427	0.7573	352.98

rate of 0.1, Nesterov momentum of 0.9 and batch size of 64. The value of the hyper-parameters was obtained by performing classification tests by varying the values of the learning rate in a range of $[1, 0.001]$, with increments of 0.01. Table 2 summarizes the resulting architectures applied to each training set; the column "Dataset" specifies the number of the training set used, column N_p specifies the number of parameters in the neural network model, column T_a specifies the percentage of classification on the training set, column V_a shows the classification percentage on the validation set obtained by that neural network model, and column T_t shows the total training and validation time. Figure 5 shows the classification accuracies for each neural network, number of classes and number of loops of each training set; showing better results for DMN over DNN models.

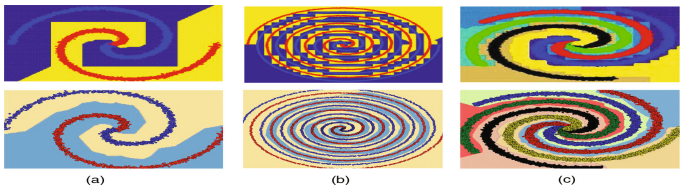

Fig. 4. Decision boundary generated by DMN (first row) and by DNN (second row).

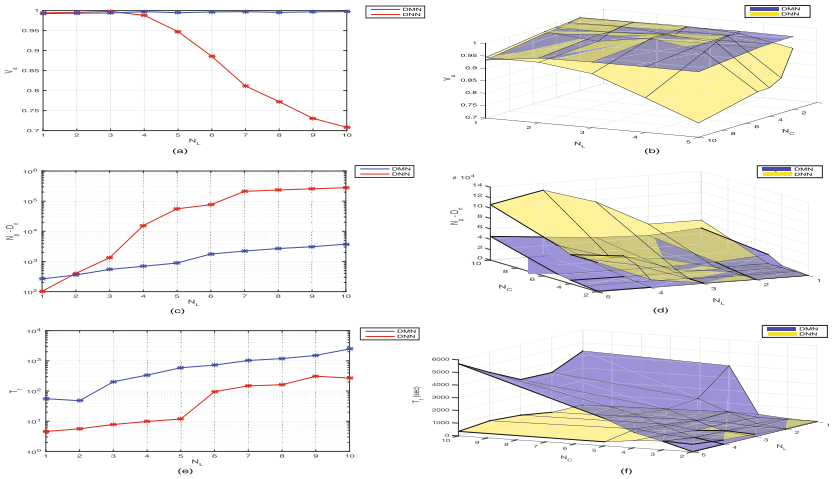

Fig. 5. (a) Classification percentages for the dataset 1 (2 classes, 10 loops), (b) classification ratios for datsets 2–5 (2–10 classes, 1–5 loops). (c) and (d) number of parameters used to classify each dataset. (e) and (f) classification times used to classify each dataset. (N_L, number of loops, N_C, number of classes).

5.3 Experimetal Results for DMNs

In the same way as in Sect. 5.2, in Table 3 the architecture of the DMN is presented; the first column shows the training set number used and the third column G_i shows the index of generalization of the DMN.

5.4 Decision Boundaries

This section compares the decision boundaries generated by the two types of neural network architectures (DNN and DMN) on the same training sets specified in Sect. 5.1. As we observe, the nature of each algorithm is very different, generating approximations to hyperplanes/hyperboxes, which yield similar results. However, for the specific dataset used, we can observe that the generation of hyperboxes of variable size best models the training set with a higher classification rate and less parameters in the DMN model. These results can be observed in Fig. 4. Each pair of images grouped by column, shows the decision

Table 3. Experimental results for DMNs.

Dataset	N_p	G_i	T_a	V_a	T_t
1.1	264	0.0066	0.9968	0.9924	4.56
1.2	356	0.0089	0.9972	0.9936	5.62
1.3	544	0.0136	0.9964	0.994	7.77
1.4	692	0.0173	0.9977	0.9972	9.84
1.5	888	0.0222	0.999	0.9948	11.83
1.6	1764	0.011	0.9974	0.9965	95.43
1.7	2216	0.0111	0.9976	0.9973	148.02
1.8	2684	0.0447	0.9978	0.9955	161.81
1.9	3080	0.0128	0.9976	0.9967	309.37
1.10	3748	0.0156	0.9987	0.9975	268.05

Dataset	N_p	G_i	T_a	V_a	T_t
2.1	2432	0.0203	0.9979	0.9836	111.62
2.2	3344	0.0111	0.9977	0.9908	315.09
2.3	2636	0.0088	0.9978	0.9935	266.36
2.4	3500	0.0097	0.9979	0.9941	388.60
2.5	3900	0.0108	0.9977	0.9946	653.01

Dataset	N_p	G_i	T_a	V_a	T_t
3.1	4244	0.0265	0.9946	0.9766	279.00
3.2	6892	0.0144	0.994	0.9837	1083.17
3.3	5944	0.0124	0.9973	0.99	954.77
3.4	6020	0.0125	0.9955	0.9897	1080.68
3.5	6864	0.0143	0.9977	0.9928	1212.82

Dataset	N_p	G_i	T_a	V_a	T_t
4.1	17632	0.0294	0.9946	0.9715	3953.38
4.2	10872	0.0181	0.9978	0.9853	1968.76
4.3	9288	0.0155	0.9954	0.9864	1714.53
4.4	10172	0.017	0.9978	0.9906	1902.82
4.5	11080	0.0185	0.9964	0.9906	2150.75

Dataset	N_p	G_i	T_a	V_a	T_t
5.1	30812	0.077	0.9921	0.9339	4335.73
5.2	24168	0.0604	0.9942	0.9616	3226.45
5.3	28024	0.0701	0.9941	0.9648	3267.14
5.4	36572	0.0914	0.9981	0.965	4417.48
5.5	45544	0.1139	0.998	0.9642	5690.73

boundary generated by the DMN (top) and the DNN (bottom). As can be seen in column (b), the decision boundaries are best defined by the DMN (column (b), top) than the decision boundaries generated by the DNN (column (b), bottom).

6 Conclusion and Future Work

Linear filters with non-linear activation functions (and back-propagation) are today the battle horses of the neural network community. This leads us to ask the questions: Are there other mathematical structures that produce better results for some problems? What advantages would they have? The motivation of this research is to answer these questions. In this paper, we compare DNNs and DMNs in a very simple 2D classification problem: multi-class spirals with increasing number of loops. We show that the performance of the DMNs surpasses that of the DNNs in terms of higher accuracies and a lesser number of learning parameters. Of course, these results are limited to spiral-like problems, which we specifically designed to test the ability of separation for the two neural architectures. It is clear that the DMN training time is longer than the DNN training time, furthermore, the classification rate is not compromised, that is, the DNNs can be trained in a shorter time, but their validation accuracy is much lower to that obtained by the DMN.

We conclude that this result is due to the nature of both algorithms. The hyperboxes of DMNs make better models for these types of datasets because the divide and conquer training is based on geometrical interpretation of the

whole data, and refines the model each recursion step, while training based on gradient descent is a search method in a dark environment only guided by partial dataset information, and local information cost function. From this, we raise the hypothesis that deep learning networks can be improved adding morphological neurons. This is a consideration for future research.

Acknowledgments. E. Zamora and H. Sossa would like to acknowledge the support provided by UPIITA-IPN and CIC-IPN in carrying out this research. This work was economically supported by SIP-IPN (grant numbers 20170836 and 20170693), and CONACYT grant number 65 (Frontiers of Science). G. Hernández acknowledges CONACYT for the scholarship granted towards pursuing his PhD studies.

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. Trans. Neur. Netw. **5**(2), 157–166 (1994)
2. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
3. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
4. Durbin, R., Rumelhart, D.E.: Product units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput. **1**(1), 133–142 (1989)
5. Giles, C.L., Maxwell, T.: Learning, invariance, and generalization in high-order neural networks. Appl. Opt. **26**(23), 4972–4978 (1987)
6. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G.J., Dunson, D.B., Dudik, M. (eds.), AISTATS, vol. 15. JMLR Proceedings, pp. 315–323 (2011). JMLR.org
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
8. Gurney, K.N.: Training nets of hardware realizable sigma-pi units. Neural Networks **5**(2), 289–303 (1992)
9. Hinton, G., Deng, L., Dong, Y., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012)
10. Hussain, A.: A new neural network structure for temporal signal processing. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1997, Munich, Germany, 21–24 April, pp. 3341–3344 (1997)
11. Ivakhnenko, A.G.: Polynomial theory of complex systems. IEEE Trans. Syst. Man Cybern. **SMC–1**(4), 364–378 (1971)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
13. LeCun, Y., Bengio, Y.: The handbook of brain theory and neural networks. In: Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. MIT Press, Cambridge (1998)
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)

15. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backprop. In: Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 1524, pp. 9–50. Springer, Heidelberg (1998). doi:10.1007/3-540-49430-8_2
16. MacKay, D.J.C.: A practical bayesian framework for backpropagation networks. Neural Comput. **4**(3), 448–472 (1992)
17. Milenkovic, Z., Obradovic, S., Litovski, V.: Annealing based dynamic learning in second-order neural networks. In: IEEE International Conference on Neural Networks, vol. 1, pp. 458–463. IEEE (1996)
18. Pessoa, L.F.C., Maragos, P.: Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition. Pattern Recogn. **33**(6), 945–960 (2000)
19. Ritter, G.X., Iancu, L., Urcid, G.: Morphological perceptrons with dendritic structure. In: The 12th IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2003, St. Louis, Missouri, USA, 25–28 May 2003, pp. 1296–1301 (2003)
20. Ritter, G.X., Urcid, G.: Lattice algebra approach to single-neuron computation. IEEE Trans. Neural Networks **14**(2), 282–295 (2003)
21. Ritter, G.X., Urcid, G.: Learning in lattice neural networks that employ dendritic computing. In: Kaburlasos, V.G., Ritter, G.X. (eds.) Computational Intelligence Based on Lattice Theory. SCI, vol. 67, pp. 25–44. Springer, Heidelberg (2007)
22. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition. In: Learning Internal Representations by Error Propagation, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
23. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Networks **61**, 85–117 (2015)
24. Sossa, H., Guevara, E.: Efficient training for dendrite morphological neural networks. Neurocomputing **131**, 132–142 (2014)
25. Van Der Malsburg, C.: Frank Rosenblatt: principles of neurodynamics: perceptrons and the theory of brain mechanisms. In: Palm, G., Aertsen, A. (eds.) Brain Theory. Springer, Heidelberg (1986)
26. Wasserman, P.D., Schwartz, T.J.: Neural networks. II. What are they and why is everybody so interested in them now? IEEE Expert **3**(1), 10–15 (1988)
27. Zamora, E., Sossa,H.: Dendrite morphological neurons trained by stochastic gradient descent. In: IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8, December 2016
28. Zamora, E., Sossa, H.: Regularized divide and conquer training for dendrite morphological neurons. In: Mechatronics and Robotics Service: Theory and Applications, Mexican Mechatronics Association, November 2016
29. Zhang, Q., Benveniste, A.: Wavelet networks. IEEE Trans. Neural Networks **3**(6), 889–898 (1992)