3-1-2019

A Dendritic Transfer Function in a Novel Fully-connected Layer

Mark Robert Musil
Portland State University

A dendritic transfer function in a novel fully-connected layer

Undergraduate Honor's Thesis

1st Mark Musil
Electrical and Computer Engineering Department
Portland State University
mmusil@pdx.edu

***Abstract*—Dendritic branch operations in pyramidal neurons are well understood in-vivo but their potential as computational assets in deep neural networks has not been explored. The pre-processing which dendrites perform may be able to decrease the error of an artificial neuron because each dendrite serves as an independent filtering mechanism which may prevent false positives. In order to test this hypothesis, a fully-connected layer implementing the dendritic transfer function is defined and used to replace the final fully-connected layer used in a standard CNN (convolutional neural network). Results show that the defined algorithm is not able to predict better than chance and possible causes are discussed. A framework for developing future dendritic layers is established.**

***Index Terms*—Dendrite, transfer function, dendritic, neural network**

I. Introduction

Modern artificial neural networks are only loosely based on the structure of the human brain but have shown great success in image recognition, functional estimation, and other tasks. The model that is widely used - although greatly customized - is the feed-forward network with the familiar equation $y = \sigma(\vec{x}W + \vec{b})$. This model uses a very simple model of the neuron as can be seen in Figure 1.

Despite this success, machine learning researchers are hesitant to return to the human brain for more insights. This hesitation from the scientific community leaves room for investigation into which, if any, processes performed by the human brain may be useful for machine learning goals.

In particular, the neuron contains many signal processing abilities that have not been fully explored. This thesis will ask the following question: can a non-linear dendritic branch function be combined with a two-layer neuron model to create a computational unit that, when used to create a fully connected layer, shows improved performance of a CNN (Convolutional Neural Network) on the MNIST data-set versus an unaltered CNN?

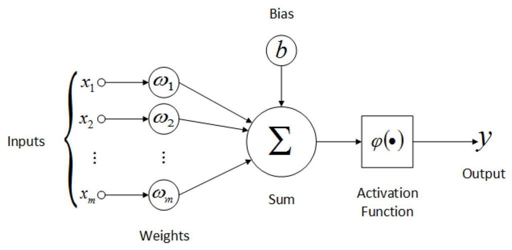

Figure 1. The classic artificial neuron, which assumes that all synapses are connected to the same lossless dendritic branch and are only acted upon after reaching the soma (cell body). Source: scielo.br

II. Background

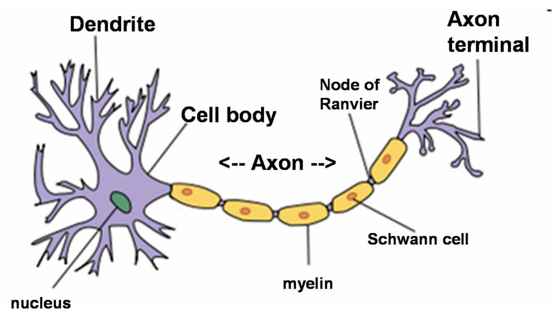

Figure 2. A simple model of the neuron. Source: biology.stackexchange.com

It has long been known that stimuli input to the spine-like synapses along the dendritic branch are not conducted directly to the soma (cell body) in all cases [1] (see Figure 2) . There are several factors that impact how much – if any – of a certain stimulus reaches the soma [2]. First, the distance from the synapse to the soma can affect the net voltage change of the soma because of leakage current that exists along the dendrite. Second, if many stimuli occur at distant locations along a branch but at roughly the same time, the soma's net voltage difference will not be the sum of the voltage changes at the synapse due to each stimulus because of saturation. The same 'muting' effect takes place when many stimuli arrive at relatively close locations but not necessarily at the same time. The aforementioned effects, modeled in this thesis, do not take into account inter-branch interactions or how different charge carriers impact signal transfer.

While the above detail is interesting, the impact on an artificial network's performance as a result of a dendritic function must be discussed. The dendrite's function as an independent threshold unit [1] is most appealing because it could prevent strong false positive neuron activations. Because each dendrite sees only a portion of the input space, each dendritic branch can confirm or deny the presence of a certain input pattern across its connections and convey this information to the soma. The soma is less likely to exhibit a (strong) false positive because the dendrites of the neuron will never have been activated.

One risk of this experiment is that dendritic thresholding could lead to false negatives. As an example, consider a neuron with 10 dendrites, one dendrite failing to recognize an input pattern could mislead the entire neuron by effectively adding noise equal to 10% of the input signal strength. In addition, using the architecture displayed in 4 may not handle spatial variance well because the dendritic receptive fields are narrow and highly local. One improvement is to implement the transfer function (or some simpler form) as part of a convolutional layer. Such complexities will be addressed more closely in future works.

The next section will place this experiment in the context of similar experiments.

III. Literature Review

The behavior of dendrites has long been studied and recognized as computationally significant. In [3] the role of the dendrite in memory is discussed and in [1] it is acknowledged that dendrites work as separate thresholding units pre-filtering the input to the soma. [2] further establishes the importance and nature of dendritic computation. [2] even discusses certain organisms who possess certain single-neuron systems capable of solving high order problems using dendritic processing. For example, the locust has a single neuron in its visual system that, using only dendritic computations, searches for objects on a collision course with the locust's head [2]. Such single-neuron tasks show that important processing takes place within dendrites.

It is no small task to translate an observed natural phenomenon into something that can be implemented in code. The first attempt at formalizing the dendrite for computation models was the sigma-pi neuron [4][5] so called because the neuron sums over input which have been multiplied together. Many of these models only include one layer of processing and could not compete with deep networks. Work was done by [6] using dendritically similar neurons using binary activations between layers. In that work, dendrites are leveraged to process binary-valued sparse distributed representations. Although [6] motivated this thesis, it was decided that a real-valued model was more appropriate.

[7] establish a 'sigma-sigma' dendritic model (used in this work) which is simpler and more similar to the artificial neurons used in deep neural networks (see Figure 1). Building on [7], the authors of [8] propose a two-layer neuron model which uses the dendritic transfer function in the first layer in lieu of the sigmoid used by [7]. Using this model is preferable because it will integrate more easily into larger networks.

There is no apparent evidence of an experiment which attempts to place this two-layer neuron model into a deep neural network and evaluate the resultant performance. It is crucial that this space is explored in order to test the hypothesis that dendrites have significance for computational problems beyond those observed in the human brain.

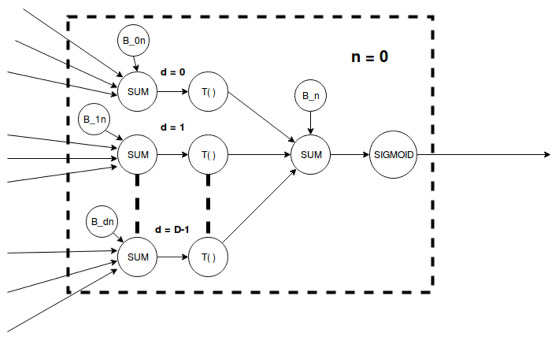

Figure 3. The schematic for the neuron model that includes the dendritic activation mechanism. $T(\vec{X})$ is the novel function that is being implemented. Two sets of summation and a final sigmoid output are also parts of the model. This figure is the author's work.

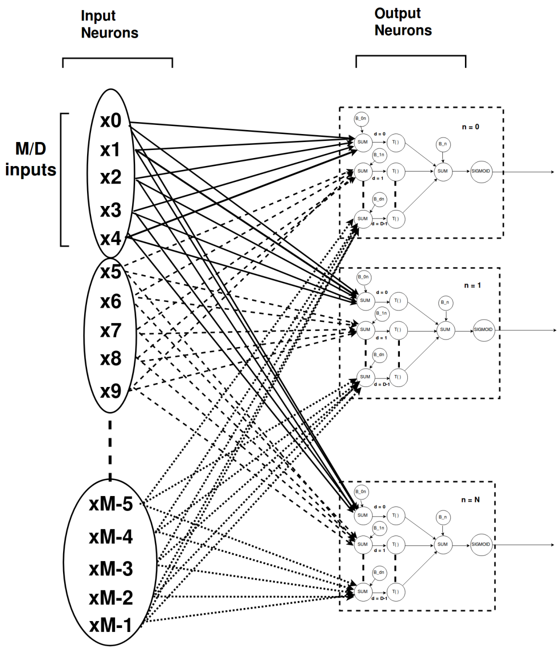

Figure 4. A limited display of the proposed fully connected layer. Note that the same group of M/D inputs connect to the same dth dendrite of each neuron. This figure is the author's work.

IV. Methods

A. Layer Architecture

The graphical representation for the dendritically modeled neuron may be seen in Figure 3. $T(Y)$ mimics the electrical qualities of a dendritic branch by normalizing groups of inputs before they reach the final summation and activation.

This dendritically inspired model will be used to create a fully connected layer that can be incorporated into existing neural networks. For these experiments the neural network into which the new layer will be placed is a CNN. CNN's are well-suited for computer vision tasks[9] and provide a good platform for testing the dendritic neuron's performance. The layer is designed to work in any feed-forward network trained using backpropagation.

Figure 4 shows the proposed structure for the novel layer while Figure 3 shows a graphic of a single dendritic neuron. It is important to notice that each dendritic branch (what the arrows are pointing to) does not see all of the input space but only the D^{th} fraction of the input space where D is the number of dendrites per neuron. The reader should note that in this novel artificial model each dendrite has mutually exclusive inputs that do not interact. This is an acceptable simplification at this stage but biological dendrites have much overlap and this will be considered in future experiments.

Once each dendrite has seen its window of the input space the soma of the neuron 'decides' whether the information that was presented to the soma (by the dendrites) matches the patterns which that neuron has been trained to recognize. The dendritic neurons will be assembled in layers of N neurons where each neuron will be fully connected to the input space but no single dendrite will see the entire input space.

B. The Transfer Function

The transfer function is split into two functions: the activation function $T(\vec{Y})$ and the boundary function $g(y)$ [8]. Although when the dendritic transfer function is discussed $T(\vec{Y})$ is the intended meaning. In addition, the multivariate logistic-sigmoid σ_{multi} is also defined in equation 1 and is used in $T(\vec{Y})$.

$$\sigma_{multi}(\vec{X}) : R^n \rightarrow R = [1 + exp(-\Sigma X_i)]^{-1} \tag{1}$$

$$g(y) = ln(\frac{(1+exp(\alpha_L(y-b_L)))^{\alpha_L^{-1}}}{(1+exp(\alpha_U(y-b_U)))^{\alpha_U^{-1}}}) + b_L \tag{2}$$

	α_L	α_U	b_U	b_L
Value	0.5	0.5	0	1
	a_d	c_d	b_d	
Value	1	Trained	”	

Table I
TABLE OF VALUES FOR $T(\vec{Y})$ AND $g(y)$.

$$T(\vec{Y_{dn}}) = g(c_d\sigma_{multi}(a_d[\vec{Y_{dn}} - b_d]) + \Sigma Y_{dn,i}) \tag{3}$$

Note that b_d is a single real number and not a vector. Table I lists the miscellaneous variables in these equations.

Note that $\vec{Y}$ and y are used as notation for the inputs to $g(y)$ and $T(\vec{Y})$ as opposed to the conventional x because the input to the dendritic transfer function is the output of the classic layer weight function $\vec{Y_{dn}} = \vec{X}W_{dn} + B_{dn}$. $\vec{X}$ is the input activation of the previous layer and B_{dn} is the bias term. W_{dn} is the weight matrix for dth dendrite in the nth neuron.

The algorithm for the layer implementation is described in algorithm 1. The algorithm assumes that **N** neurons and **D** dendrites per neuron have already been chosen for the layer in which the computation is taking place. The subscripts d and n denote individual dendrites (d) or neurons (n). B_{dn} (two subscripts) is the bias for a single dendrite while B_n (one subscript) is the bias for the soma of the neuron. ϕ is the activation function (such as a standard sigmoid, ReLU, etc.) to be put after the soma, and O_n is the output of the nth neuron in the output layer.

Algorithm 1: The algorithm that defines the functionality of the novel layer.

```
Data: Set of input activations from the
      previous layer $\vec{X}$
Result: Set of output activations of the
        current layer $\vec{O}$
1 for $n \leftarrow 0$ to $N-1$ do
2     for $d \leftarrow 0$ to $D-1$ do
3         Compute $\vec{Y_{dn}} = \vec{X}W_{dn} + B_{dn}$ ;
4         Compute $T(\vec{Y_{dn}})$;
5     end
6     Compute
        $O_n = \phi([\sum_{j=0}^{j=D-1} T(\vec{Y_{jn}})] + B_n)$;
7 end
```

The algorithm will be implemented as a network layer following the structure displayed in figure 4. This will allow the robust and approachable Keras interface to be leveraged during the project. The completed source code is on GitHub [10] where it is publicly accessible under the GNU General Public License v3.0.

V. RESULTS

The layer was tested as part of a fully connected network trained on the Iris dataset. The network was trained against another fully connected control network which had the same configuration as the test network. Both networks have input and output dimensions of 4 and 3, respectively. The control network had hidden layer of 5 units. The test network had two hidden layers, one of 6 units and the other being the dendritic layer which was before the output layer. The dendritic layer had 3 neurons, each with two dendrites, each dendrite receiving input from 3 neurons of the previous layer.

It was found that the dendritic layer network never performed better than chance on classification. This was consistent across different sets of hyper-parameters (learning rate range: [0.001 0.04], epochs: [50 - 500]). Error vs epoch graphs for each network are shown in Figures 5 and 6.

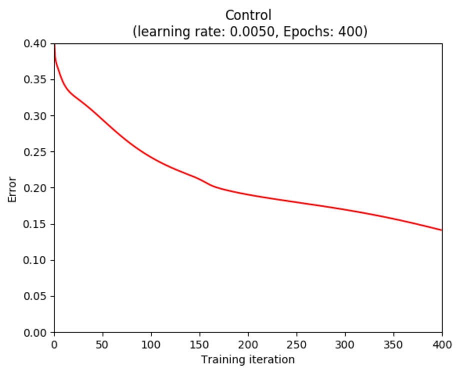

Figure 5. Results from the control Iris network. Hidden layer size is 5 neurons.

VI. DISCUSSION AND FUTURE WORK

After a post-experiment analysis, it was found that the network did not converge because of the

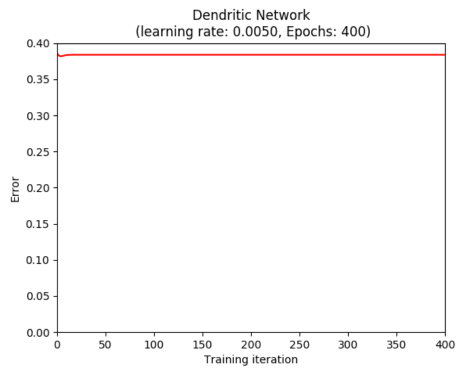

Figure 6. Test results using the added dendritic layer.

architecture chosen in the dendritic layer. The dendrites in each neuron were connected in an identical manner to the dendrites in every other neuron. The weights were also initialized in the same way for each dendrite, meaning that each neuron in the dendritic layer ended up yielding a very similar output.

Although there was some variation seen during early epochs, (one run in particular showed that the output was [0.56, 0.51, 0.551] before any training) The output activation tended towards [0.5, 0.5, 0.5] and largely remained there throughout training. As this is a classification problem, this means that the network had no confidence in any particular class being the correct answer. This explains the flat-line in Figure 6.

It is hypothesized that the problem comes from the network architecture and specifically the homogeneity of the connections to the dendrites, which may results in a structure that cannot create decision boundaries. For future experiments, a novel set of connections should be explored. Potentially a randomized set of connections may be advantageous. Otherwise it may be good to only allow each neuron to view a 'window' of the input space and within this window randomize the connections.

The dendrite's role is clearly important to the way in which neuron's process information. There are many interesting facets of a dendrite's function and some animals even rely on single neuron systems which leverage dendrites to make decisions. In addition, the potential reduction in false positives for neurons is an exciting promising result. The formalism established here will allow for future successful experiments into dendrites and their unique behavior.

The most significant accomplishment of this thesis then is the addition of a formality for approaching dendritically inspired artificial neuron models. This will pave the way for further experiments in this domain and allow future missteps to be avoided.

References

[1] S. Sardi, R. Vardi, A. Goldental, and I. Kanter, "New types of experiments reveal that a neuron functions as multiple independent threshold units," *Neuron*, vol. 7, no. 18036, 2017.

[2] M. London and M. Hausser, "Dendritic computation," *Annual Review of Neuroscience*, vol. 28, 2005.

[3] P. Poirazi and B. W. Mel, "Impact of active dendrites and structural plasticity on the memory capacity of neural tissue," *Neuron*, vol. 29, no. 5451, p. 779–796, 2001.

[4] J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cognitive Science*, vol. 6, no. 3, pp. 205–254, 1982.

[5] Y. Shin and J. Ghosh, "The pi-sigma network: an efficient higher-order neural network for pattern classification and function approximation," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. i, July 1991, pp. 13–18 vol.1.

[6] S. Ahmad and J. Hawkins, "How do neurons operate on sparse distributed representations? A mathematical theory of sparsity, neurons and active dendrites," jan 2016. [Online]. Available: https://arxiv.org/abs/1601.00720

[7] P. Poirazi, B. T., and B. W. Mel, "Pyramidal neuron as two-layer neural network," *Neuron*, vol. 37, no. 6, pp. 989–99, 2003.

[8] M. F. Singh and D. H. Zald, "A simple transfer function for non-linear dendritic integration," *Frontiers in Computational Neuroscience*, vol. 9, no. 10, 2015.

[9] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 329–344.

[10] M. Musil. (2018) Github repository for this honor's thesis. [Online]. Available: https://github.com/mmusil25/CorGraphProject/tree/master/DendriticLayer