

Might a Single Neuron Solve Interesting Machine Learning Problems Through Successive Computations on Its Dendritic Tree?

Ilenna Simone Jones

ilennaj@pennmedicine.upenn.edu

*Department of Neuroscience, University of Pennsylvania,
Philadelphia, PA 19104, U.S.A.*

Konrad Paul Kording

kording@upenn.edu

*Departments of Neuroscience and Bioengineering, University of Pennsylvania,
Philadelphia, PA 19104, U.S.A.*

Physiological experiments have highlighted how the dendrites of biological neurons can nonlinearly process distributed synaptic inputs. However, it is unclear how aspects of a dendritic tree, such as its branched morphology or its repetition of presynaptic inputs, determine neural computation beyond this apparent nonlinearity. Here we use a simple model where the dendrite is implemented as a sequence of thresholded linear units. We manipulate the architecture of this model to investigate the impacts of binary branching constraints and repetition of synaptic inputs on neural computation. We find that models with such manipulations can perform well on machine learning tasks, such as Fashion MNIST or Extended MNIST. We find that model performance on these tasks is limited by binary tree branching and dendritic asymmetry and is improved by the repetition of synaptic inputs to different dendritic branches. These computational experiments further neuroscience theory on how different dendritic properties might determine neural computation of clearly defined tasks.

1 Introduction

1.1 Dendritic Nonlinearities. Though the role of biological neurons as the mediators of sensory integration and behavioral output is clear (Jones & Kording, 2019), the computations performed within neurons have been a point of investigation for decades (McCulloch & Pitts, 1943; Hodgkin & Huxley, 1952; FitzHugh, 1961; Poirazi, Brannon, & Mel, 2003b; Mel, 2016). For example, the McCulloch and Pitts (M&P) neuron model is based on an approximation that a neuron linearly sums its input and maps this through a nonlinear threshold function, allowing it to carry out a selection of

logic-gate-like functions that can be expanded to create logic-based circuits (McCulloch & Pitts, 1943). The M&P neuron also sets the foundation for neurons in artificial neural networks (ANNs), where each neuron in the network linearly sums its input and maps this through a nonlinear activation function (Goodfellow, Bengio, & Courville, 2016; LeCun, Bengio, & Hinton, 2015). ANNs, made up of often millions of these neurons, are demonstrably powerful algorithms that can be trained to solve complex problems, from reinforcement learning to natural language processing to computer vision (LeCun, Bengio, & Hinton, 2015; Krizhevsky, Sutskever, & Hinton, 2006; Mnih et al., 2015; Devlin, Chang, Lee, & Toutanova, 2018; Huval et al., 2015). However, M&P neurons and neurons of ANNs are point-neuron models that rely on linear sums of their inputs, whereas the observed physiology of biological neurons shows that dendrites impose nonlinearities on their synaptic inputs before summation at the soma (London & Häusser, 2005; Poirazi, Brannon, & Mel, 2003a; Antic, Zhou, Moore, Short, & Ikonomu, 2010; Agmon-Snir, Carr, & Rinzel, 1998). This indicates that M&P and ANN neurons may radically underestimate what individual neurons can do.

It is known that dendritic nonlinearities are responsible for a variety of neuronal dynamics and can be used to mechanistically explain the roles of biological neurons in a variety of behaviorally significant circuits (London & Häusser, 2005; Agmon-Snir et al., 1998; Barlow & Levick, 1965). For example, passive properties of dendrites lead to attenuation of current along the dendrite, allowing for low-pass filtering of inputs (London & Häusser, 2005; Rall, 1959). Active properties of dendrites allow for synaptic clustering (Mel, 1993) to result in superlinear summation of voltage inputs upon reaching the soma (Antic et al., 2010; Schiller, Major, Koester, & Schiller, 2000; Branco & Häusser, 2011; Tran-van minh, Cazé, Abrahamsson, Gutkin, & Digregorio, 2015). These properties allow important functions such as auditory coincidence detection and even logical operations within dendrites (Mel, 2016; London & Häusser, 2005; Agmon-Snir et al., 1998; Koch, Poggio, & Torre, 1983). To fully explore the scope of biological neuron function, it is important to model more sophisticated computations within dendritic trees.

Models for individual neurons with meaningful dendrites have been proposed to better understand neuron computation (Mel, 2016; Gerstner & Naud, 2009). Biologically detailed approaches, such as employing the multicompartmental biophysical model (Hines & Carnevale, 1997), have been fitted to empirical data in order to study dendritic dynamics such as backpropagating action potentials and nonlinear calcium spikes (London & Häusser, 2005; Hay, Hill, Schürmann, Markram, & Segev, 2011; Wilson, Whitney, Scholl, & Fitzpatrick, 2016). Poirazi et al. (2003b) pioneered a more abstracted approach of modeling single neurons that isolates the impacts of including the biological detail of dendritic sigmoidal nonlinearities on predicting neural firing rates produced by neuronal biophysical models with dendrites. Though the main takeaway from this paper was that dendritic

nonlinearity was likely to be sigmoidal, the nuance of this novel approach was that the architecture of this two-layer ANN was sparsely connected in a way that was analogous to the sparse, binary tree connections of a dendrite. The biological property of binary tree sparsity implemented in this model established that it is possible to model individual neurons as ANNs. Importantly, ANNs can be optimized to compute high-dimensional input-output functions like a many-synaptic input to a biophysical neuron, resulting in a firing rate output. This opens the possibility for investigation of higher-dimensional input to neuron models and the sparse morphological context within which they are located.

The relationship between the location of active synapses and their somatic output has been investigated (Schiller et al., 2000; Poirazi & Mel, 2001; Tran-van minh et al., 2015; London & Häusser, 2005), but the biological property of morphological tree-like sparsity of connections has not. Poirazi's two-layer ANN has the sparsity, but the focus of that paper was not on the model's tree-like sparsity as a biological property to be investigated. Also, unlike Poirazi's two-layer ANN, it should be noted that dendrites are very asymmetric, where many synapses or "leaves" off the tree can be present on an unbranched dendrite. Moreover, barring the work coming out of Idan Segev's group (David, Idan, & Michael, 2019; Moldwin & Segev, 2019; Moldwin, Kalmenson, & Segev, 2020), very few of these studies have asked about the kinds of machine learning problems that such neurons could solve. How dendritic sparse binary tree constraint and asymmetry affect neural computation, and thus potential performance on real machine learning problems, thus deserves an investigation.

1.2 Repetition of Synaptic Inputs. While the morphology of a dendritic tree is key to modeling its computational capabilities (Mel, 1993, 2016; London & Häusser, 2005; Segev, 2006; Wilson et al., 2016), it may also be important to consider the role of repeated synaptic inputs to the same postsynaptic neuron. Complex computation in ANNs depends on dense connection, which repeats inputs to each node in each layer (LeCun et al., 2015). Empirically, electron microscopy studies have shown that a presynaptic axon synapses approximately four times per postsynaptic neuron (Kincaid, Zheng, & Wilson, 1998). Also, these studies show evidence of a certain kind of repeated synapses called multisynaptic boutons (MSBs) (Jones, Klintsova, Kilman, Sirevaag, & Greenough, 1997). MSBs have been shown to occur 11.5% of the time in rats living in enriched environments (Jones et al., 1997). It has been shown that an *in vitro* long-term potentiation (LTP) induction protocol can also increase the number of MSBs of the same dendrite six-fold (Jones et al., 1997). LTP, involved in learning and memory (Bliss & Lomo, 1973; Stuchlik, 2014), can then lead to the replication of synapses between two neurons. In addition, it has been speculated that multisynaptic boutons are involved in improving the efficacy of neuron-to-neuron connections (Federmeier, Kleim, & Greenough, 2002), as well as

being involved in reorganization of network connectivity (Lee et al., 2013). This suggests that repeated synapses may be important for changing the computations a single neuron can do.

1.3 Contribution of This Letter. By training and testing ANNs on complex tasks, the field of machine learning gains insights into which architectures can solve which kinds of tasks (Goodfellow et al., 2016; LeCun et al., 2015). At the moment, the field of computational neuroscience lacks insights into which functions can be computed by neurons with dendrite, despite the fact that we can describe the different behaviorally significant functions individual neurons are able to fulfill (London & Häusser, 2005; Agmon-Snir et al., 1998; Barlow & Levick, 1965; Gidon et al., 2020). If we are to consider a neuron as an input/output device with a binary tree as its dendritic tree, we may be able to test its ability to learn to perform high-dimensional tasks and gain insight into how dendritic trees may affect the computation of a well-defined task.

It is important to emphasize here that mathematical models like the M&P model, the Hodgkin-Huxley conductance model, and Poirazi's two-layer ANN are all neuron models because they include representations of empirical biological details of neurons (Brette, 2015). As a result, these have more explanatory power that pertains to such details (Brette, 2015). We hold that we can theoretically investigate dendritic biological details, such as the tree structure of dendrites, by formalizing them in mathematical models, manipulating these details, and observing the outputs to learn the impacts of these manipulations.

Here we design a trainable neuron model with a dendritic tree to test its performance on binary classification tasks taken from the field of machine learning. The model comprises a sparse ANN: a binary tree in which each nonlinear unit receives only two inputs. This model was tested as a completely symmetrical, balanced tree and as a biologically plausible asymmetric tree to investigate the impacts of dendritic binary tree sparsity and asymmetry, respectively. The model also allows us to test the impact of repeated inputs on task performance. The effects of these biological properties in this abstracted neuron model were compared to a linear classifier (a lower bound) and a two-layer fully connected ANN (an upper bound). We found that our binary tree models, representing a single biological neuron, predictably performs better than a comparable linear classifier but worse than a comparable fully connected ANN. Furthermore, when repeated inputs are incorporated into our model, it often approximately matches the performance of a comparable two-layer fully connected ANN. Surprisingly, asymmetry, which introduces depth, parameters, and more nonlinearities to the model, seems to limit model performance when compared to the shallower balanced tree. These results illuminate the impacts of biological properties found in a dendritic tree and contribute theoretical insight into how these properties affect dendritic computation.

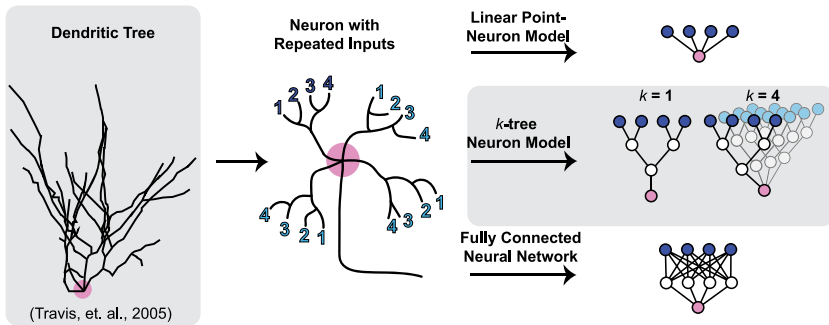


Figure 1: Novel ANN neuron model with repeated inputs. Left: Traced morphology of the dendrite of a human BA18 (occipital) cell (Travis, Ford, & Jacobs, 2005). Soma location is marked in pink. Middle: A representation of a hypothetical neuron. Inputs in dark blue at the terminal ends of one subtree are repeated in light blue in three other subtrees. Upper right: Representation of an M&P-like linear point neuron model. Middle right: k -tree neuron model, where k = number of subtrees. Each input and hidden node has leaky ReLU activation function, and the output node has a sigmoid activation function. Bottom right: Representation of a two-layer fully connected neural network (FCNN). Each input and hidden node has a leaky ReLU activation function, and the output node has a sigmoid activation function.

2 Results

One of the classical questions in neuroscience is how dendrite structure and the various synaptic inputs to the dendritic tree affect computation (London & Häusser, 2005; Mel, 2016; Rall, 1959). Traditional neuron models are designed to best match observed neural dynamics (Poirazi et al., 2003b; Gerstner & Naud, 2009; Brette et al., 2011; Gouwens et al., 2018; Hay et al., 2011; Ahrens, Huys, & Paninski, 2006), however, with exceptions (Poirazi et al., 2003b; Ujfalussy, Makara, Branco, & Lengyel, 2015; Gidon et al., 2020; Zador, Claiborne, & Brown, 1992; Zador & Pearlmuter, 1996; Legenstein & Maass, 2011), the impacts of nonlinearities and, especially, the impacts of repeated inputs on the computational capabilities of neurons have yet to be quantified in the way we suggest. The computational abilities of ANNs can be judged by their performance on various complex tasks (Goodfellow et al., 2016; LeCun et al., 2015). Following this lead, we imposed dendritic binary tree structural constraints (see Figure 1) on a trainable nonlinear ANN, resulting in a special case of sparsely connected ANN. We call this a 1-tree because it is analogous to the structure of a single soma-connected subtree of a dendritic tree (see Figure 1). By repeating this subtree structure multiple times and feeding each the exact same input, we create what we call a k -tree, where k is the number of repeated trees connected to a soma node.

Finally, it is important to point out that this conception of the k -tree is a balanced, symmetrical tree. To move closer to biological realism, we also test subtrees that are asymmetrical, whereby a leafnode of the asymmetric tree can be analogous to a synapse on a dendritic branch. By using a trainable k -tree, or an asymmetric k -tree (A- k -tree), that has a biological binary tree structure constraint and repeated inputs, we can quantitatively judge the computational performance of this neuron model on performing complex tasks.

Neurons, arguably, produce binary outputs (presence or absence of an action potential; Hodgkin & Huxley, 1952). Therefore, to fairly judge an individual neuron model's performance on a complex task, we will use a binary classification task. The complexity in the tasks can come from high-dimensional vector inputs from images taken from classic computer vision data sets used in the field of machine learning (see Figure 2).

As controls for performance comparison, we used a linear discriminant analysis (LDA) linear classifier to approximate the performance of a linear point neuron model and a fully connected neural network (FCNN) that is comparable in size to the symmetrical k -tree. The linear classifier model is relatively simple compared to the more parameter-complex k -tree and FCNN, and we expect it to be able to learn fewer functions (Dreiseitl & Ohno-Machado, 2002); therefore, its performance sets an expected lower bound. The FCNN is densely connected and consists of two layers. With its nonlinearities, we expect it to learn to express a greater variety of functions than the linear classifier; therefore, its performance sets an expected upper bound. To compare the two ANNs, let us say that n is the number of pixel inputs to each classifier, determining the number of parameters, P , needed in each network, and h is the number of nodes in the hidden layer of the FCNN. Based on the constraints of each network, the FCNN will then have $P = h(n + 1)$, and the k -tree will have $P = k(2n - 1)$. To match the number of parameters of the FCNN to that of the k -tree, we assert that $h = 2k$ (see Table 1).

2.1 Nonlinear Tree Neuron Model Performs Better Than a Linear Classifier. Classical models of neurons have been of linear point neurons that do not take into consideration dendritic nonlinearities (McCulloch & Pitts, 1943; Hodgkin & Huxley, 1952; FitzHugh, 1961). By considering dendritic nonlinearity and structure, we designed a new neuron model: a nonlinear ANN with the binary tree structural constraints of a dendritic tree called a 1-tree. We then compared the performance of this new model against a proxy for a point neuron, an LDA linear classifier. Focusing on one simple image classification task of a binary data set of handwritten numbers, MNIST, we compare the computational performance of the 1-tree and the linear classifier. Significantly, the performance of the 1-tree is greater than that of the linear classifier with $p < 0.0001$ (see Figure 3A and Table 2). Repeating this test with six more data sets, we find that for most of these

Binary Classification across tasks


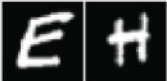
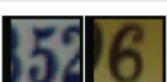
Dataset	Classes	Examples	1-D Input Size
MNIST	3, 5		32x32 = 1024
Fashion-MNIST (FMNIST)	0, 6		32x32 = 1024
EMNIST	14, 17		32x32 = 1024
Kuzushiji-MNIST (KMNIST)	2, 6		32x32 = 1024
CIFAR-10	3, 5		3x32x32 = 3072
Street View House Numbers (SVHN)	5, 6		3x32x32 = 3072
USPS	3, 5		16x16 =256

Figure 2: Classification task data set. We considered seven machine learning data set of varying content and size, each with 10 classes. For each data set, two of the classes were chosen by selecting the least linearly separable pair using a linear discriminant analysis (LDA) linear classifier. Each image was vectorized in order to be compatibly presented to each model.

Table 1: ANN Parameter Size Comparison.

<i>k</i>	256 inputs		1024 inputs		3072 inputs	
	<i>k</i> -tree	FCNN	<i>k</i> -tree	FCNN	<i>k</i> -tree	FCNN
1	511	514	2,047	2050	6,143	6,146
2	1,022	1,028	4,094	4,100	12,286	12,292
4	2,044	2,056	8,188	8,200	24,572	24,584
8	4,088	4,112	16,376	16,400	49,144	49,168
16	8,176	8,224	32,752	32,800	98,288	98,336
32	16,352	16,448	65,504	65,600	196,576	196,336

Note: Fully connected neural network (FCNN) architectures are matched in parameter size to the *k*-tree architectures.

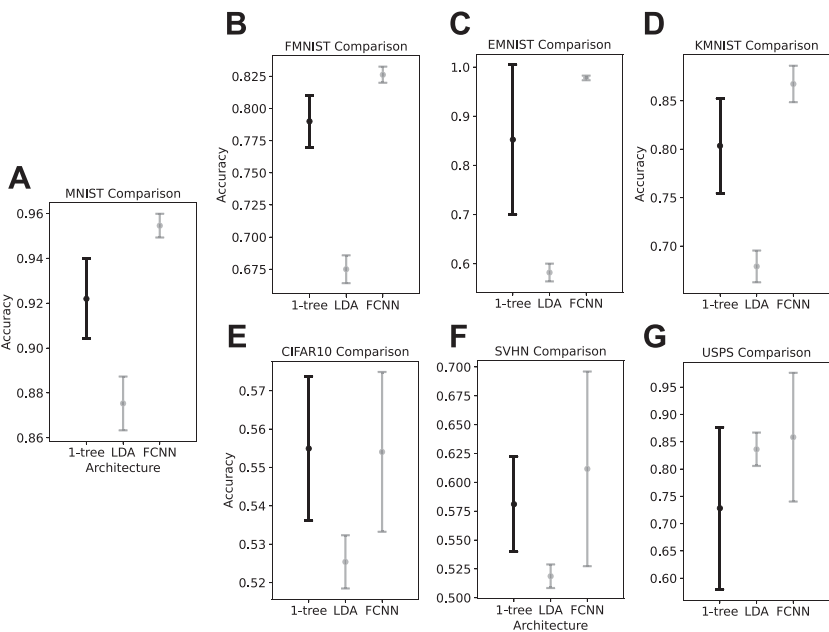


Figure 3: Performance of 1-tree compared to linear classifier and FCNN. 1-tree performance is compared to that of a lower bound, LDA, and an upper bound, FCNN. For most tasks, the 1-tree performs better than LDA, and FCNN performs better than the 1-tree.

data sets, regardless of the size of the input dimensionality in each data set, the 1-tree performs consistently above the linear classifier (FMNIST, EMNIST, KMNIST, CIFAR10: $p < 0.0001$; SVHN: $p = 0.0005$) (see Figures 3B–3F and Table 2). Exceptionally, the performance of the 1-tree on the USPS data set had no significant difference of performance compared to the linear classifier. The USPS data set has the smallest input dimensionality (256 pixels) and leads to a 1-tree with the fewest parameters ($P = 511$; see Table 1). It could be that the network was not complex enough to perform better than the linear classifier. Barring this exception, not only do dendrites have nonlinear properties, nonlinearities in a dendrite-like neuron model generally improve its computational performance compared to that of a linear classifier.

For comparison to the 1-tree, we tested a two-layer fully connected neural network (FCNN) matched in parameter size to the 1-tree. In the MNIST task, the FCNN performed significantly better than the 1-tree with a $p = 0.0001$ (see Figure 3A and Table 2). We then tested the six additional data sets, resulting in different-sized 1-trees and FCNNs due to differences in input sizes. The similar-sized FMNIST, EMNIST, and KMNIST data set

Table 2: *k*-Tree Mean Performance Comparison to FCNN and LDA.

	MNIST	FMNIST	EMNIST	KMNIST
1-tree	0.9220 ± 0.0179	0.7900 ± 0.0202	0.8524 ± 0.1520	0.8035 ± 0.0488
32-tree	0.9635 ± 0.0043	0.8300 ± 0.0063	0.9851 ± 0.0029	0.8791 ± 0.0113
A-32-tree (<i>n</i> = 1)	0.9111	0.7995	0.9402	0.8200
LDA	0.8753 ± 0.0120	0.6750 ± 0.0108	0.5821 ± 0.0180	0.6790 ± 0.0164
1-FCNN	0.9546 ± 0.0053	0.8262 ± 0.00063	0.9779 ± 0.0046	0.8674 ± 0.0188
32-FCNN	0.9696 ± 0.0053	0.8290 ± 0.0075	0.9846 ± 0.0034	0.9088 ± 0.0080
1-tree vs LDA	<i>p</i> < 0.0001	<i>p</i> < 0.0001	<i>p</i> < 0.0001	<i>p</i> < 0.0001
1-tree vs 1-FCNN	<i>p</i> = 0.0001	<i>p</i> = 0.0001	<i>p</i> = 0.0235	<i>p</i> = 0.0018
32-tree vs 32-FCNN	<i>p</i> = 0.0156	<i>p</i> = 0.7516	<i>p</i> = 0.7357	<i>p</i> < 0.0001
	CIFAR10	SVHN	USPS	
1-tree	0.5605 ± 0.0140	0.5811 ± 0.0412	0.8221 ± 0.0465	
32-tree	0.5784 ± 0.0111	0.6036 ± 0.0661	0.8981 ± 0.0080	
A-32-tree (<i>n</i> = 1)	0.5050	0.4533	0.7822	
LDA	0.5254 ± 0.0069	0.5186 ± 0.0102	0.8362 ± 0.0306	
1-FCNN	0.5592 ± 0.0148	0.6117 ± 0.0844	0.8971 ± 0.0199	
32-FCNN	0.5654 ± 0.0104	0.7794 ± 0.0301	0.9067 ± 0.0169	
1-tree vs LDA	<i>p</i> < 0.0001	<i>p</i> = 0.0005	<i>p</i> = 0.4897	
1-tree vs 1-FCNN	<i>p</i> = 0.8736	<i>p</i> = 0.4024	<i>p</i> = 0.0012	
32-tree vs 32-FCNN	<i>p</i> = 0.0344	<i>p</i> < 0.0001	<i>p</i> = 0.2031	

Notes: Performance accuracy is listed as mean ± standard error for a set of 10 trials. *p*-Values calculated using the Student *t*-test. LDA and FCNN are used as lower and upper bounds that the *k*-tree is compared to.

networks maintained the significant difference between the 1-tree and FCNN (see Figures 3B–3D and Table 2). The USPS data set also maintained a significant difference (see Figure 3G). The CIFAR10 and SVHN data sets did not have a significant difference in performance (see Figures 3E and 3F and Table 2). The high variance in the FCNN performance for CIFAR10 and SVHN (see Figures 3E and 3F and Table 2) may be due to the FCNN’s failure to train in some trials, resulting in performances close to 50%. For most tasks we tried, the FCNN performed much better than the 1-tree.

2.2 Repeating Inputs to Tree Model Increases Performance Comparable to FCNN with a Similar Number of Parameters. The computational impact of repeated inputs to a dendritic tree is not clear; however, studies have shown increased repetition of inputs as a result of plasticity events (Toni, Buchs, Nikonenko, Bron, & Muller, 1999), which has implications for learning and memory. By repeating the 1-tree structure and input to the model *k* times, we can then achieve a *k*-tree neuron model (see Figure 1). This can be a proxy for seeing how repeated inputs might affect computational performance on various binary image classification tasks. Returning to the MNIST data set, we tested *k* = 1, 2, 4, 8, 16, 32 and observed how

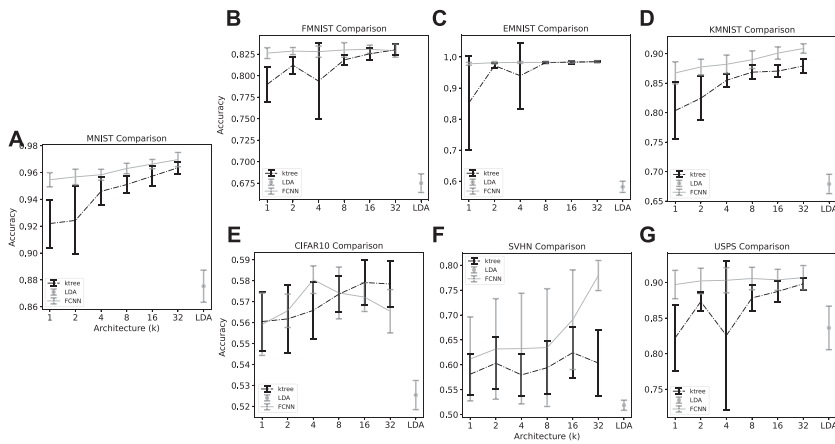


Figure 4: Performance of a k -tree compared to a linear classifier and FCNN. k -tree performance is compared to that of a lower bound, LDA, and an upper bound, FCNN. The k is doubled five times, resulting in tests of $k = 1, 2, 4, 8, 16, 32$. In all cases, as k (the number of repeated dendritic subtrees) increases, so does the performance accuracy of the k -tree, approaching the upper bound.

increasing k can gradually improve performance. For example, compare the performance of a 1-tree to that of a 32-tree in the MNIST binary classification task ($p = 0.0000$) (see Figure 4A). Remarkably, the performance of the 32-tree ($96.35 \pm 0.43\%$) is very close to that of the FCNN ($96.96 \pm 0.53\%$) yet still different with $p = 0.0156$ (see Figure 4A and Table 2). Increasing the number of repeats to a k -tree neuron model improves its performance on the MNIST binary classification task, nearly meeting the performance of a comparable FCNN.

In order to see if this result generalizes, we tested the k -tree on six additional binary image classification data sets. All tasks see an increase in performance as the number of subtrees in the k -tree increases up to $k = 32$ (see Figures 4B and 4G). The 32-tree meets the performance of the FCNN in the FMNIST ($p = 0.7516$), EMNIST ($p = 0.7357$), and USPS ($p = 0.2031$) tasks (see Figures 4B, 4C, and 4G and Table 2). For the CIFAR10 data set, FCNN performance peaks at $k = 4$, then decreases, resulting in the 32-tree surpassing the 32-FCNN (see Figure 4E and Table 2). We can then say that increasing the number of repeats to a k -tree neuron model improves its computational performance in all tasks such that it approaches the performance of a comparable FCNN.

Finally, to investigate the impact of dendritic asymmetry (Farhoodi & Kording, 2018), we generated a binary asymmetric tree structure with approximately 256 leaves, 1024 leaves, and 3072 leaves in order to feed each tree an appropriate image input for each data set. We found that even after

repeating each asymmetric subtree k times to form A - k -trees, the performance of these models was even lower than that of the symmetrical k -trees, even when accounting for 32 input repetitions (see Table 2). Dendritic trees being asymmetric is a salient aspect of biology. And yet we find that performance is considerably worse. There are reasons to believe that this is due to the difficulty in training. As neural networks become deeper, there are more vanishing gradients (Bhumbra, 2018). With a tree being much sparser in connections than fully connected networks, it is much easier to run into vanishing gradient problems. As such, it seems likely that with gradient descent, our trees are simply hard to train. This raises an interesting question: If neurons implement nonlinear deep computational trees: how can they learn?

3 Methods

3.1 Computational Tasks. Knowing that the output of a neuron is binary (presence or absence of an action potential), we chose to train our neuron model on a binary classification task. Using standard, high-dimensional, computer vision data sets, we used a linear discriminant analysis (LDA) linear classifier to determine which two classes within each data set were least linearly separable through training the LDA linear classifier and testing it on pairs of classes (see Figure 2). We used MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao, Rasul, & Vollgraf, 2017), EMNIST (Cohen, Afshar, Tapson, & Van Schaik, 2017), Kuzushiji-MNIST (Clanuwat et al., 2018), CIFAR-10 (Krizhevsky, 2009), Street View House Numbers (SVHN) (Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2014), and USPS (Hastie, Tibshirani, & Friedman, 2001) data sets.

3.2 Controls. The controls we use are the LDA linear classifier and a fully connected neural network (FCNN). The linear classifier sets a baseline performance for linear separability of each of the two classes per data set, in addition to acting as a proxy for a linear point neuron model. The two-layer FCNN is a comparable reference to see if k -tree performance meets or exceeds that of a densely connected network. The hidden layer of the FCNN is equal to twice the number of trees ($2k$) in the k -tree it is compared to, and its output layer has one node.

3.3 Data Preprocessing. We used data sets from the torchvision (version 0.5.0) Python package. We then padded the 28 by 28 resolution images with zeros so that they were 32×32 , and flattened the images to one-dimensional vectors. We then split the shuffled training set into training and validation sets (for MNIST, the ratio was 1:5 so as to let the validation set size match the test set), Then we split the resultant shuffled training set and validation set into 10 independent subsets. Each set was used for a different cross-validation trial.

3.4 Model Architecture. Using PyTorch (version 1.4.0), we designed the k -tree model architecture to be a feedforward neural network with sparse binary-tree connections. The weight matrices, which were dense tensors, of each layer, were sparsified such that each node receives two inputs and produces one output. For example, the 1024 pixel-size images were fed to a 1-tree with 10 layers: the input layer is 1024 by 512, the second layer 512 by 256, and so on until the penultimate layer is reached with dimensions 2 by 1. The final layer is k by 1 where k is the number of subtrees in the k -tree; in this case, it would be 1 by 1. In the special case of the 3072 pixel size images, inputs were fed into a 1-tree with 11 layers, the input layer is 3072 by 1024, the second layer is 1024 by 512, and so on. The A - k -tree structures were generated using a Markov chain Monte Carlo generator, whereby all trees were made using the same parameters. The trees were chosen if the number of leaves they had were close enough to 256, 1024, and 3072. This way, we can compare symmetric binary trees with asymmetric ones.

To account for the sparsification, we altered the initialization of the weight matrices: we used standard “Kaiming normal” initialization with a gain of 1/density of sparsified dense tensor weight matrices. We also created a “freeze mask” that recorded which weights were set to zero in order to freeze those weights during training later. All nonzero weights were trained in model optimization. For the forward step, we used leaky ReLU with a slope of 0.01 for nodes between layers, and sigmoid nonlinearity at the final output node, which kept output values between zero and one.

3.5 Model Training. The model, inputs, and labels were loaded onto a Nvidia GeForce 1080 GPU using CUDA version 10.1. The batch size was 256. Early stopping was used such that after 60 epochs where no decrease in the validation loss is observed, training is stopped. Loss was calculated using binary cross-entropy loss. We used an Adam optimizer with a learning rate of 0.001. Within the training loop, immediately after the backward step and before updating the weights using the gradients, we zeroed out the gradients indicated by the freeze mask so as to keep the model sparsely connected. Each train-test loop was run for 10 trials with a different training subset each trial and the same test set every trial. Trial averages and standard deviation were then calculated, and p -values were calculated using Student t -test.

4 Discussion

Here we quantify the potential computational capabilities of an abstracted neuron model with dendritic features and repeated inputs. We designed a trainable neuron model aimed at imitating computation within a dendrite: a sparse ANN with binary constraints made up of nonlinear nodes (see Figure 1). The tree that resulted from this constraint was repeated k times with identical inputs in order to explore the impacts of repeated inputs. We

quantified model performance on seven high-dimensional binary image classification tasks (see Figure 2) and compared performance to a linear classifier, a lower bound, and a comparable FCNN, arguably an upper bound. In this context, we see the tree-like structure of real dendrites as a constraint that potentially makes computation harder. The 1-tree, with its nonlinear nodes and dendritic structure constraint, generally performed better than the linear classifier (see Figure 3). When we increased k of the k -tree from $k = 1$ to $k = 32$, we saw a consistent increase in k -tree performance across all tasks (see Figure 4). In the case of the MNIST task, the performance of the 32-tree was close to the comparable FCNN performance. Surprisingly, the 32-tree in the FMNIST, EMNIST, and USPS tasks met that of the comparable FCNN. Interestingly, the asymmetric A- k -tree largely underperformed the k -tree. The mathematical neuron model we implemented with its approximation to real dendritic trees illuminates the impacts of tree structures of a dendrite, asymmetry, and input repetition on dendritic computation of high-dimensional tasks.

A limitation of this study is that our mathematical neuron model is analogous to a real dendrite only by having certain biological properties of the dendrite, namely, tree structure and input repetition. This model of a neuron obviously does not comprehensively include all biological details of neurons. However, introducing each biological detail to a mathematical model that performs tasks allows us to clearly observe the impact of those details on computation. As with all neuron models, there is some level of abstraction to the model that allows the modeler to clearly manipulate and discern how particular biological details lead to effective prediction of neuronal outputs. Methodically investigating the impacts of dendritic details on computation contributes a basis for further theoretical development.

Another limitation of this study is the relevance of our computational tasks. Although it is hard to know exactly what kind of input a neuron receives from its presynaptic connections, we do not believe the one-dimensional unstructured input we provide our neuron model is biologically plausible. Ordering the pixel input to these models randomly overall decreases k -tree performance (see Figure S1 in the supplementary material), implying that the order of the input affects performance. Further investigation may be needed to explore how the ordering of the one-dimensional pixel input might affect performance. In biology, this ordering itself is likely learnable through bouton turnover.

The binary tree structure we chose to constrain our model to make the k -tree makes several assumptions. The leaves and branches of the tree are analogous to synaptic inputs and dendritic branches, respectively, whereby in the model, the weights for each type of node are either synaptic input weights or axial resistance weights. In biological dendritic trees, each compartment will receive an exclusive set of inputs. Therefore, we chose not to use convolution or any kind of weight sharing in our model. In addition, in our model, the synaptic weights and internode weights are real-valued

free parameters; however, the weights would be more biologically realistic if there were limited to positive scalar values. (Rall, 1959; Huys, Ahrens, & Paninski, 2006) Future work to address this would be to constrain the free parameter ranges to be nonnegative. Finally, in an attempt to make the k -trees more realistic, we tested the A - k -trees, which can have many leaves, or synapses, on a long stretch of many dendritic branch compartments. This generally made these asymmetric trees far deeper. We found that the A - k -tree performance was much lower than the balanced k -tree. This may be because the nature of an asymmetric tree makes this very sparse structure much deeper and thus much more difficult to train. Further work can be done to troubleshoot the engineering of this kind of sparse model. Importantly, though, it raises an interesting question. If biological dendrites are deep trees, how do they manage to learn successfully? Gradient descent may not be the last word about training deep trees.

It is important to acknowledge that the repetition in the structure of a k -tree necessarily increases the number of parameters in the model. We can then expect that as the number of parameters increases in a model, the higher its performance should be. However, this illustrates what may be happening when real synaptic input repetitions occur on different dendritic branches. Adding more degrees of freedom by including more dendritic compartments or “parameters” in a real neuron when repeating the inputs is analogous to adding parameters in our model. Therefore, our model makes the important theoretical step in explaining the role of repeated dendritic inputs to different dendrites in expanding neural computation. Notably, the FCNN control has fewer nonlinearities than the k -tree and is an architecturally much shallower network. However, the FCNN controls for how density (and thereby sparsity) affects computation, while matching the k -tree in parameter size. Controlling for both the number of nonlinearities and density at the same time would yield such a control incomparable due to the large number of parameters it would require. To have some form of upper-bound control, we then decided on using a simple FCNN that is parameter size-matched, even though it may be limited in comparability to the k -tree.

The development of training algorithms for binary trees may also be somewhat relevant for the field of deep learning. The k -trees we consider are special cases of sparse ANN, wherein there are only two inputs to all nodes after the first layer. These contrast with randomly made sparse networks or pruned sparse networks (Frankle & Carbin, 2019) because they have very severe constraints. It is then surprising that a k -tree could perform at the level of a comparable FCNN. We would be interested in future work to compare the performance of binary tree structures, inspired by biological dendrites, against the performance of less structured sparse ANNs with comparable edge density. If we could figure out how to train deep binary trees, this finding may be relevant to new approaches to the development of hardware specialized to neural networks.

This study tests the classification performance of a model of computation on dendrites and compares it to a model that follows the point-neuron assumption, highlighting the importance of considering branching dendrite structure and nonlinearities when modeling neurons. We expand this test to consider the possibility of repeated synaptic inputs in our model, showing that the model consistently performs better with more repeated inputs to additional subtrees. We also see that the symmetrical tree-like network neuron model we designed can reach performance similar to that of a densely connected network, while the asymmetrical version of this model has limited performance. Fundamentally, this study is a foray into directly considering a neuron's computational capability by training the model to perform complex tasks using deep learning methodology, which promises to further our insights into single neuron computation.

Acknowledgments

We thank the members of the Kording Lab, specifically Roozbeh Farhooi, Ari Benjamin, and David Rolnick for their help over the development of this project.

Code

The code for this project can be found at the following github repository: <https://github.com/ilennaj/ktree>

References

- Agmon-Snir, H., Carr, C. E., & Rinzel, J. (1998). The role of dendrites in auditory coincidence detection. *Nature*, 393, 268–272.
- Ahrens, M. B., Huys, Q. J. M., & Paninski, L. (2006). Large-scale biophysical parameter estimation in single neurons via constrained linear regression. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems*, 18. Cambridge, MA: MIT Press.
- Antic, S. D., Zhou, W. L., Moore, A. R., Short, S. M., & Ikonomu, K. D. (2010). The decade of the dendritic NMDA spike. *Journal of Neuroscience Research*, 88(14), 2991–3001.
- Barlow, H. B., & Levick, W. R. (1965). The mechanism of directionally selective units in rabbit's retina. *Journal of Physiology*, 178(3), 477–504.
- Bhumbra, G. S. (2018). *Deep learning improved by biological activation functions*. arXiv:1804.11237. (pp. 1–9).
- Bliss, T. V. P., & Lomo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology*, 232(2), 357–374.
- Branco, T., & Häusser, M. (2011). Synaptic integration gradients in single cortical pyramidal cell dendrites. *Neuron*, 69(5), 885–892.

- Brette, R. (2015). What is the most realistic single-compartment model of spike initiation? *PLOS Computational Biology*, 11(4), 1–13.
- Brette, R., Fontaine, B., Magnusson, A. K., Rossant, C., Platkiewicz, J., & Goodman, D. F. M. (2011). Fitting neuron models to spike trains. *Frontiers in Neuroscience* 5(February), 1–8.
- Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., & Ha, D. (2018). Deep learning for classical Japanese literature. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*, 31 (pp. 1–8). Red Hook, NY: Curran.
- Cohen, G., Afshar, S., Tapson, J., & Van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 2921–2926). Piscataway, NJ: IEEE.
- David, B., Idan, S., & Michael, L. (2019). *Single cortical neurons as deep artificial neural networks*. bioRxiv:613141.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pretraining of deep bidirectional transformers for language understanding. arXiv:1810.04805.
- Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: A methodology review. *Journal of Biomedical Informatics*, 35(5–6), 352–359.
- Farhoodi, R., & Kording, K. P. (2018). *Sampling neuron morphologies*. bioRxiv. <https://doi.org/10.1101/248385>.
- Federmeier, K. D., Kleim, J. A., & Greenough, W. T. (2002). Learning-induced multiple synapse formation in rat cerebellar cortex. *Neuroscience Letters*, 332(3), 180–184.
- FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6), 445–466.
- Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proceedings of the 7th International Conference on Learning Representations* (pp. 1–42).
- Gerstner, W., & Naud, R. (2009). How good are neuron models? *Science*, 326(5951), 379–380.
- Gidon, A., Zolnik, T. A., Fidzinski, P., Bolduan, F., Papoutsi, A., Poirazi, P., . . . Larkum, M. E. (2020). Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473), 83–87.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, MA: MIT Press.
- Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Proceedings of the 2nd International Conference on Learning Representations* (pp. 1–13).
- Gouwens, N. W., Berg, J., Feng, D., Sorensen, S. A., Zeng, H., Hawrylycz, M. J., . . . Arkhipov, A. (2018). Systematic generation of biophysically detailed models for diverse cortical neuron types. *Nature Communications*, 9(1).
- Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer-Verlag.
- Hay, E., Hill, S., Schürmann, F., Markram, H., & Segev, I. (2011). Models of neocortical layer 5b pyramidal cells capturing a wide range of dendritic and perisomatic active properties. *PLOS Computational Biology*, 7(7).

- Hines, M. L., & Carnevale, N. T. (1997). The NEURON simulation environment. *Neural Computation*, 9(6), 1179–1209.
- Hodgkinx & Huxley (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiology*, 1117, 500–544.
- Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., . . . Ng, A. Y. (2015). *An empirical evaluation of deep learning on highway driving*. arXiv:1504.01716.
- Huys, Q. J. M., Ahrens, M. B., & Paninski, L. (2006). Efficient estimation of detailed single-neuron models. *Journal of Neurophysiology*, 96(2), 872–890.
- Jones, I. S., & Kording, K. P. (2019). Quantifying the role of neurons for behavior is a mediation question. *Behavioral and Brain Sciences*, 42, E233.
- Jones, T. A., Klintsova, A. Y., Kilman, V. L., Sirevaag, A. M., & Greenough, W. T. (1997). Induction of multiple synapses by experience in the visual cortex of adult rats. *Neurobiology of Learning and Memory*, 68(1), 13–20.
- Kincaid, A. E., Zheng, T., & Wilson, C. J. (1998). Connectivity and convergence of single corticostriatal axons. *Journal of Neuroscience* 18(12), 4722–4731.
- Koch, C., Poggio, T., & Torre, V. (1983). Nonlinear interactions in a dendritic tree: Localization, timing, and role in information processing. In *Proceedings of the National Academy of Sciences of the United States of America*, 80, 2799–2802.
- Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images* (Technical Report TR-2009). Toronto: University of Toronto.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2006). ImageNet classification with deep convolutional neural networks. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems*, 18 (pp. 713–772). Cambridge, MA: MIT Press.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 86(11).
- Lee, K. J., Park, I. S., Kim, H., Greenough, W. T., Pak, D. T., & Rhyu, I. J. (2013). Motor skill training induces coordinated strengthening and weakening between neighboring synapses. *Journal of Neuroscience*, 33(23), 9794–9799.
- Legenstein, R., & Maass, W. (2011). Branch-specific plasticity enables self-organization of nonlinear computation in single neurons. *Journal of Neuroscience*, 31(30), 10787–10802.
- London, M., & Häusser, M. (2005). Dendritic computation. *Annual Review of Neuroscience*, 28(1), 503–532.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4), 115–133.
- Mel, B. (2016). Toward a simplified model of an active dendritic tree. In G. J. Stuart, N. Spruston, N., & M. Häusser, (Eds.), *Dendrites*. Oxford Scholarship Online.
- Mel, B. W. (1993). Synaptic integration in an excitable dendritic tree. *Journal of Neurophysiology*, 70(3), 1086–101.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Moldwin, T., Kalmenson, M., & Segev, I. (2020). *The gradient clusteron: A model neuron that learns via dendritic nonlinearities, structural plasticity and gradient descent*. bioRxiv. <https://doi.org/10.1101/2020.12.15.417790>.

- Moldwin, T., & Segev, I. (2019). *Perceptron learning and classification in a modeled cortical pyramidal cell*. bioRxiv:464826.
- Poirazi, P., Brannon, T., & Mel, B. W. (2003a). Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell. *Neuron*, 37(6), 977–987.
- Poirazi, P., Brannon, T., & Mel, B. W. (2003b). Pyramidal neuron as two-layer neural network. *Neuron*, 37(6), 989–999.
- Poirazi, P., & Mel, B. W. (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron*, 29(3), 779–796.
- Rall, W. (1959). Physiological properties of dendrites. *Annals of the New York Academy of Sciences*, 96(4), 1071–1092.
- Schiller, J., Major, G., Koester, H. J., & Schiller, Y. (2000). NMDA spikes in basal dendrites. *Nature*, 1261(1997), 285–289.
- Segev, I. (2006). What do dendrites and their synapses tell the neuron? *Journal of Neurophysiology*, 95(3), 1295–1297.
- Stuchlik, A. (2014). Dynamic learning and memory, synaptic plasticity and neurogenesis: An update. *Frontiers in Behavioral Neuroscience*, 8), 1–6.
- Toni, N., Buchs, P., Nikonenko, I., Bron, C. R., & Muller, D. (1999). LTP promotes formation of multiple spine synapses between a single axon terminal and a dendrite. *Nature*, 402, 421–425.
- Tran-van minh, A., Cazé, R. D., Abrahamsson, T., Gutkin, B. S., & Digregorio, D. A. (2015). Contribution of sublinear and supralinear dendritic integration to neuronal computations. *Frontiers in Cellular Neuroscience*, 9, 1–15.
- Travis, K., Ford, K., & Jacobs, B. (2005). Regional dendritic variation in neonatal human cortex: A quantitative Golgi study. *Developmental Neuroscience*, 27(5), 277–287.
- Ujfalussy, B. B., Makara, J. K., Branco, T., & Lengyel, M. (2015). Dendritic nonlinearities are tuned for efficient spike-based computations in cortical circuits. *eLife*, 4, 1–51.
- Wilson, D. E., Whitney, D. E., Scholl, B., & Fitzpatrick, D. (2016). Orientation selectivity and the functional clustering of synaptic inputs in primary visual cortex. *Nature Neuroscience*, 19(8), 1003–1009.
- Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms*. arXiv:1708.07747.
- Zador, A. M., Claiborne, B. J., & Brown, T. H. (1992). Nonlinear pattern separation in single hippocampal neurons with active dendritic membrane. In J. Moody, S. J. Hanson, & R. Lippmann (Eds.), *Advances in neural information processing systems*, 4 (pp. 51–58). San Mateo, CA: Morgan Kaufmann.
- Zador, A. M., & Pearlmutt, B. A. (1996). VC dimension of an integrate-and-fire neuron model. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory* (pp. 10–18). New York: ACM.