Non-linear Neurons with Human-like Apical Dendrite Activations

Mariana-Iuliana Georgescu [1], Radu Tudor Ionescu [2], Nicolae-Catalin Ristea [1], and Nicu Sebe [1]

[1]Affiliation not available
[2]University of Bucharest

October 30, 2023

Non-linear Neurons with Human-like Apical Dendrite Activations

Mariana-Iuliana Georgescu [1] **Radu Tudor Ionescu** [1] **Nicolae-Cătălin Ristea** [2] **Nicu Sebe** [3]

Abstract

In order to classify linearly non-separable data, neurons are typically organized into multi-layer neural networks that are equipped with at least one hidden layer. Inspired by some recent discoveries in neuroscience, we propose a new neuron model along with a novel activation function enabling learning of non-linear decision boundaries using a single neuron. We show that a standard neuron followed by the novel apical dendrite activation (ADA) can learn the XOR logical function with 100% accuracy. Furthermore, we conduct experiments on three benchmark data sets from computer vision and natural language processing, i.e. Fashion-MNIST, UTKFace and MOROCO, showing that the ADA and the leaky ADA functions provide superior results to Rectified Liner Units (ReLU) and leaky ReLU, for various neural network architectures, e.g. 1-hidden layer or 2-hidden layers multi-layer perceptrons (MLPs) and convolutional neural networks (CNNs) such as LeNet, VGG, ResNet and Character-level CNN. We also obtain further improvements when we change the standard model of the neuron with our pyramidal neuron with apical dendrite activations (PyNADA).

1. Introduction

The power of neural networks in classifying linearly non-separable data lies in the use of multiple (at least two) layers. We take inspiration from the recent study of Gidon et al. (2020) in order to propose a simpler yet more effective approach: a new computational model of the neuron, termed pyramidal neuron with apical dendrite activations (PyNADA), along with a novel activation function, termed apical dendrite activation (ADA), allowing us to classify linearly non-separable data using an individual neuron.

[1]University of Bucharest, Bucharest, Romania [2]Politehnica University of Bucharest, Bucharest, Romania [3]University of Trento, Trento, Italy. Correspondence to: Radu Tudor Ionescu <raducu.ionescu@gmail.com>.

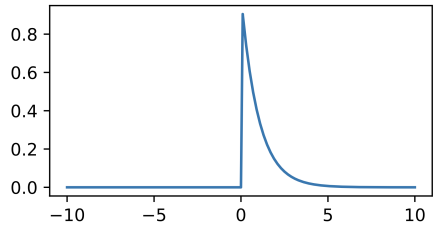

Figure 1. Activation function observed in apical dendrites of pyramidal neurons in the human cerebral cortex. The input corresponds to the horizontal axis and the output to the vertical axis.

Biological motivation. Recently, Gidon et al. (2020) observed that the apical dendrites of pyramidal neurons in the human cerebral cortex have a different activation function than what was previously known from observations on rodents. The newly-discovered apical dendrite activation function produces maximal amplitudes for electrical currents close to threshold-level stimuli and dampened amplitudes for stronger electrical currents, as shown in Figure 1. This new discovery indicates that an individual pyramidal neuron from the human cerebral cortex can classify linearly non-separable data, contrary to the conventional belief that non-linear problems require multi-layer neural networks. This is the main reason that motivated us to propose ADA and PyNADA.

Psychological motivation. Remember the first time you ate your favorite dish. Was it better than the second or the last time you ate the same dish? According to Knutson and Cooper (2006), our brains provide higher responses to novel stimuli than known (repetitive) stimuli. This means that our brain gets bored while eating the same dish over and over again, although it might become our favorite dish. If we were to model the brain response over time for a certain stimulus, we would obtain the function illustrated in Figure 1. This is yet another reason to propose and experiment with ADA and PyNADA.

Mathematical motivation. Despite the recent significant advances in various application domains (Xu et al., 2017; Wang et al., 2018) brought by deep learning (LeCun et al., 2015), state-of-the-art deep neural networks rely on an old and simple mathematical model of the neuron introduced by Rosenblatt (1958). Minsky and Papert (1969) argued that a single artificial neuron is incapable of learning non-linear functions, such as the XOR function. In order to classify

linearly non-separable data, standard artificial neurons are typically organized in multi-layer neural networks that are equipped with at least one hidden layer. Contrary to the common belief, we propose an activation function (ADA) that transforms a single artificial neuron into a non-linear classifier. We also prove that the non-linear neuron can learn the XOR logical function with 100% accuracy. Hence, the ADA function can increase the computational power of artificial neurons.

Empirical motivation. We provide empirical evidence in favor of replacing the commonly-used activation functions, such as Rectified Liner Units (ReLU) (Nair & Hinton, 2010) and leaky ReLU (Maas et al., 2013), with the ones proposed in this work, namely ADA and leaky ADA, in various neural network architectures ranging from 1-hidden layer or 2-hidden layers multi-layer perceptrons (MLPs) to convolutional neural networks (CNNs) such as LeNet (LeCun et al., 1998), VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016) and Character-level CNN (Zhang et al., 2015). We attain accuracy improvements on several tasks: object class recognition on Fashion-MNIST (Xiao et al., 2017), gender prediction and age estimation on UTKFace (Zhang et al., 2017) and Romanian dialect identification on MOROCO (Butnaru & Ionescu, 2019). We report further accuracy improvements when the standard artificial neurons are replaced with our pyramidal neurons with apical dendrite activations.

In summary, our contribution is threefold:

- We propose a new artificial neuron called pyramidal neurons with apical dendrite activations (PyNADA) along with a new activation function called apical dendrite activation (ADA).
- We demonstrate that, due to the novel apical dendrite activation, a single neuron can learn the XOR logical function.
- We show that the proposed ADA and PyNADA provide superior results to standard neurons based on the notorious ReLU and leaky ReLU activations. In most cases, our improvements are statistically significant.

2. Related Work

Activation functions. Since activation functions have a large impact on the performance of deep neural networks (DNNs), studying and proposing new activation functions is an interesting topic (Hayou et al., 2019). Nowadays, perhaps the most popular activation function is ReLU (Nair & Hinton, 2010). Formally, ReLU is defined as $max(0, x)$, where x is the input. Because ReLU is linear on the positive side (for $x > 0$), its derivative is 1, so it does not saturate like *sigmoid* and *tanh*. On the negative side of the domain, ReLU is constant (equal to 0), so the gradient is 0. Hence, a neuron that uses ReLU as activation function cannot update its weights via gradient-based methods on examples for which the neuron is inactive. If we have neurons that never activate in a neural network, we risk not being able to train the neural model using gradient-based methods. To eliminate the problem caused by inactivate neurons with ReLU activation, Maas et al. (2013) introduced leaky ReLU. The leaky ReLU function is defined as:

$$y = ReLU_{leaky}(x, l) = l \cdot \min(0, x) + \max(0, x), \quad (1)$$

where l is a number between 0 and 1 (typically very close to 0), allowing the gradient to pass even if $x < 0$. While in the leaky version of ReLU the parameter l is kept fixed, He et al. (2015) proposed Parametric Rectified Linear Units (PReLU), in which the leak l is learned by back-propagation. Different from ReLU, the Exponential Linear Unit (ELU) (Clevert et al., 2016) outputs negative values, while still avoiding the vanishing gradient problem on the positive side of the domain. This helps in bringing the mean unit activation down to zero, enabling faster convergence times. Another generalization of ReLU is the Maxout unit (Goodfellow et al., 2013), which instead of applying an element-wise function, divides the input into groups of k values and then outputs the maximum value of each group.

In contrast to all the recent (ReLU, PReLU, ELU, etc.) and historically-motivated (*sign*, *sigmoid*, *tanh*, etc.) activation functions, we propose an activation function that transforms a single artificial neuron into a non-linear classifier. To support our statement, in Section 3, we prove that a neuron followed by our apical dendrite activation function can learn the XOR logical function.

Models of artificial neurons. To our knowledge, the first mathematical model of the biological neuron is the perceptron (Rosenblatt, 1958). The perceptron solves linearly separable problems by computing a linear combination of the input and the weights, then adding the bias. The result is passed through the *sign* transfer function, obtaining the final output. The Rosenblatt's perceptron was introduced along with a rule for updating the weights, which converges to a solution only if the data set is linearly separable. Although the perceptron is a simple and old model, it represents the foundation of modern DNNs. Different from the Rosenblatt's perceptron, the Adaptive Linear Neuron (ADALINE) (Widrow, 1960) updates its weights via stochastic gradient descent, back-propagating the error before applying the sign function. ADALINE has the same disadvantage as Rosenblatt's perceptron, namely that it cannot produce non-linear decision boundaries. Stochastic gradient descent is currently the most popular algorithm of training DNNs, although some researchers agree that it is not the best choice, lacking biological motivation (Lee et al., 2015).

In contrast to Rosenblatt's perceptron and ADALINE, we propose an artificial neuron that has two input branches, the basal branch and the apical tuft. The apical tuft is par-

ticularly novel because it uses a novel activation function (Gidon et al., 2020) that can solve non-linearly separable problems.

Although there are a few works that studied various aspects of the modeling of pyramidal neurons, e.g. segregated dendrites in the context of deep learning (Guerguiev et al., 2017) or memorizing sequences with active dendrites and multiple integration zones (Hawkins & Ahmad, 2016), to our knowledge, we are the first to propose an artificial pyramidal neuron that is inspired by the recent discovery of Gidon et al. (2020). Different from previous studies, the apical tuft of our pyramidal neuron is equipped with the novel apical dendrite activation suggested by Gidon et al. (2020). Furthermore, we integrate our pyramidal neuron into various deep neural architectures, showing its benefits over standard artificial neurons. We note that Gidon et al. (2020) have not presented the pyramidal neuron in a computational scenario, hence we are the first modeling it computationally.

Other less related neural models are Kohonen Self-Organizing Maps (SOMs) (Kohonen, 1990) and Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). Both SOM and LSTM models are specific types of neural architectures able to solve only certain tasks. We however propose a novel artificial neuron that can be plugged into different neural architectures, solving a broad range of tasks.

3. Non-Linear Neurons

3.1. ADA: Apical Dendrite Activation Function

The activation function illustrated in Figure 1 and deduced from the experiments conducted by Gidon et al. (2020) can be formally expressed as follows:

$$y = \begin{cases} 0, & \text{if } x < 0 \\ exp(-x), & \text{if } x \geq 0 \end{cases}, \quad (2)$$

where $x \in \mathbb{R}$ is the input of the activation function and y is the output. Since the function defined in Equation (2) is not useful in practice[1], we propose a closed form definition that approximates the activation function defined in Equation (2), as follows:

$$y = ADA(x, \alpha, c) = \max(0, x) \cdot exp(-x \cdot \alpha + c), \quad (3)$$

where $x \in \mathbb{R}$ is the input of the activation function, $\alpha > 0$ is parameter that controls the width of the peak, $c > 0$ is a constant that controls the height of the peak and y is the output. Similar to ReLU, our apical dendrite activation (ADA) is saturated on the negative side, i.e. its gradients are equal to zero for $x < 0$. Thus, a neural model trained with back-propagation (Rumelhart et al., 1986) would not update the corresponding weights. We therefore propose leaky ADA, a more generic version that avoids saturation on the negative side, just as leaky ReLU. We formally extend the definition of ADA from Equation (3) to leaky ADA as follows:

$$y = ADA_{leaky}(x, \alpha, c, l) = l \cdot \min(0, x) + ADA(x, \alpha, c), \quad (4)$$

where $0 \leq l \leq 1$ is the leak parameter controlling the function steepness on the negative side and the other parameters are the same as in Equation (3). By setting $l = 0.005$, $\alpha = 1$ and $c = 1$ in Equation (4), we obtain the activation function illustrated in Figure 2. By comparing Figures 1 and 2, we observe that the (leaky) ADA function defined in Equation (4) has a similar shape to the transfer function defined in Equation (2). Indeed, both functions have no activation or almost no activation when $x < 0$. Then, there is a high activation peak for small but positive values of x. Finally, the activation damps along the horizontal axis, as x gets larger and larger.

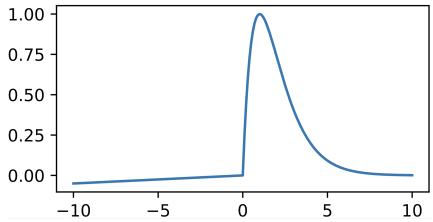

Figure 2. Our leaky apical dendrite activation (ADA) function that can be expressed in closed form (see Equation (4)). The input corresponds to the horizontal axis and the output to the vertical axis. This output is obtained by setting $l = 0.005$, $\alpha = 1$ and $c = 1$ in Equation (4).

Lemma 1. *There exists an artificial neuron followed by the apical dendrite activation from Equation (3) which can predict the labels for the XOR logical function, by rounding its output.*

Proof. Given an input data sample represented as a row vector $x \in \mathbb{R}^n$, the output $y \in \mathbb{R}$ of an artificial neuron with ADA is obtained as follows:

$$y = ADA(x \cdot w + b, \alpha, c), \quad (5)$$

where α and c are defined as in Equation (3), w is the column weight vector and b is the bias term. The following equation shows how to obtain the rounded output:

$$y = \lfloor ADA(x \cdot w + b, \alpha, c) \rceil, \quad (6)$$

where $\lfloor \cdot \rceil$ is the rounding function. Similarly, we can obtain the rounded outputs for an entire set of data samples represented as row vectors in an input matrix X:

$$Y = \lfloor ADA(X \cdot w + b, \alpha, c) \rceil. \quad (7)$$

Let X and T represent the data samples and the targets

[1] Commonly-used deep learning frameworks such as TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019) do not cope well with functions containing *if* branches.

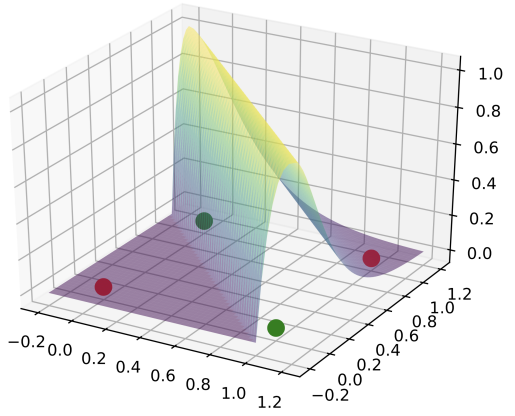

Figure 3. The output of a neuron with apical dendrite activation, as defined in Equation (5), able to classify the XOR logical function. The output is obtained by setting the weights to $w = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$, the bias term to $b = -4$ and the parameters of the ADA function to $\alpha = 1$ and $c = 1$. Large output values (closer to 1) correspond to the green data points labeled as class 1, while low output values (closer to 0) correspond to the red data points labeled as class 0. Best viewed in color.

corresponding to the XOR logical function, i.e.:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \; T = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}. \tag{8}$$

We next provide an example of weights and parameters to prove our lemma. By setting $w = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$, $b = -4$, $\alpha = 1$ and $c = 1$ in Equation (7), we obtain the following output:

$$\begin{aligned} Y &= \left\lfloor ADA\left(\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 5 \end{bmatrix} - 4, 1, 1 \right) \right\rfloor \\ &= \left\lfloor ADA\left(\begin{bmatrix} -4 \\ 1 \\ 1 \\ 6 \end{bmatrix}, 1, 1 \right) \right\rfloor \approx \left\lfloor \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0.04 \end{bmatrix} \right\rfloor = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}. \end{aligned} \tag{9}$$

Since the output Y computed in Equation (9) is equal to the target T defined in Equation (8), it results that Lemma 1 is true. □

Our proof is intuitively explained in Figure 3. The four data points from the XOR data set are represented on a plane and the output of the neuron followed by ADA is represented on the axis perpendicular to the plane in which the XOR points reside. The output for the red points (labeled as class 0) is 0 or close to 0, while the output for the green points (labeled as class 1) is 1. Applying the rounding function $\lfloor \cdot \rceil$ on top of the output depicted in Figure 3 is equivalent to setting a threshold equal to 0.5, labeling all points above the threshold with class 1 and all points below the threshold with class 0. This gives us the labels for the XOR logical function.

Corollary 2. *There exists an artificial neuron followed by the apical dendrite activation from Equation (3) which can predict the labels for the OR logical function, by rounding its output.*

Proof. We can trivially prove Corollary 2 by following the proof for Lemma 1. We just have to set α and c to different values, e.g. $\alpha = 0.4$ and $c = 0.5$. □

Corollary 3. *There exists an artificial neuron followed by the apical dendrite activation from Equation (3) which can predict the labels for the AND logical function, by rounding its output.*

Proof. We can trivially prove Corollary 3 by following the proof for Lemma 1. We just have to set the bias term to a different value, e.g. $b = -9$. □

From Lemma 1, Corollary 2 and Corollary 3, it results that an artificial neuron followed by ADA has more computational power than a standard artificial neuron followed by sigmoid, ReLU or other commonly-used activation functions. More precisely, the ADA function enables individual artificial neurons to classify both linearly and non-linearly separable data.

3.2. PyNADA: Pyramidal Neurons with Apical Dendrite Activations

Pyramidal neurons have two types of dendrites: apical dendrites and basal dendrites. Electrical impulses are sent to the neuron through both kinds of dendrites and the impulse is passed down the axon, if an action potential occurs. Prior to Gidon et al. (2020), it was thought that apical and basal dendrites had identical activation functions. This is because experiments were usually conducted on pyramidal neurons extracted from rodents. In this context, proposing an artificial pyramidal neuron would not make much sense, because its mathematical model would be identical to a standard artificial neuron. Gidon et al. (2020) observed that the apical dendrites of pyramidal neurons in the human cerebral cortex have a different (previously unknown) activation function, while the basal dendrites exhibit the well-known hard-limit transfer function. This observation calls for a new model of artificial pyramidal neurons. We therefore propose pyramidal neurons with apical dendrite activations (PyNADA). Given an input data sample $x \in \mathbb{R}^n$, the output $y \in \mathbb{R}$ of a PyNADA is obtained through the following equation:

$$y = ReLU(x \cdot w' + b') + ADA(x \cdot w'' + b'', \alpha, c), \tag{10}$$

where α and c are defined as in Equation (3), w' and w'' are column weight vectors and b' and b'' are bias terms. A

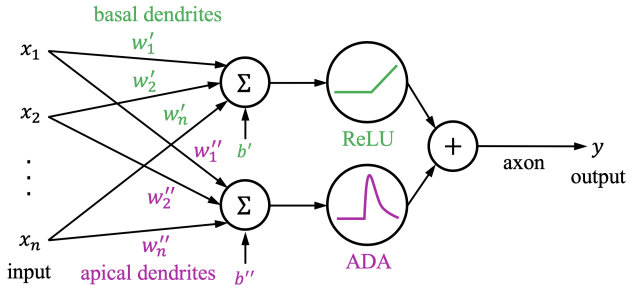

Figure 4. A pyramidal neuron with apical dendrite activations (PyNADA). The input x goes through the basal dendrites followed by ReLU and through the apical tuft followed by ADA. The results are summed up and passed through the axon. Best viewed in color.

graphical representation of PyNADA is provided in Figure 4. In the proposed model, the input x is distributed to the basal dendrites represented by the weight vector w' and the bias term b', and to the apical dendrites (apical tuft) represented by the weight vector w'' and the bias term b''.

For practical reasons, we replace the hard-limit transfer function, suggested by Gidon et al. (2020) for the basal dendrites, with the ReLU activation. This change ensures that we can optimize the weights w' and the bias b' through backpropagation, i.e. we have at least some non-zero gradients. Since the intensity of electrical impulses is always positive, the biological model proposed by Gidon et al. (2020) is defined for positive inputs and the thresholds for the activation functions are well above 0. However, an artificial neuron can also take as input negative values, i.e. $x \in \mathbb{R}^n$. Hence, the thresholds of the activation functions used in PyNADA are set to 0.

Corollary 4. *There exists a pyramidal neuron with apical dendrite activation, as defined in Equation (10), which can predict the labels for the XOR logical function, by rounding its output.*

Proof. We can trivially prove Corollary 4 by following the proof for Lemma 1. For the apical tuft, we can simply set $w'' = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$, $b'' = -4$, $\alpha = 1$ and $c = 1$ in Equation (10), just as in the proof for Lemma 1. The demonstration results immediately when we simply drop out the basal dendrites by setting $w' = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $b' = 0$. □

4. Experiments

4.1. Data Sets

We conduct experiments on three data sets: Fashion-MNIST (Xiao et al., 2017), UTKFace (Zhang et al., 2017) and MOROCO (Butnaru & Ionescu, 2019).

Fashion-MNIST (Xiao et al., 2017) is a recently introduced data set that shares the same structure as the more popular MNIST (LeCun et al., 1998) data set, i.e. it contains 60,000 training images and 10,000 test images that belong to 10 classes of fashion items. We use a subset of 10,000 images from the training set for validation. The size of each image is 28×28 pixels. Fashion-MNIST was essentially proposed because the accuracy on MNIST has saturated to nearly 100% on the test set, thus being hard to report any further improvements.

UTKFace (Zhang et al., 2017) is a large-scale data set of 23,708 images. The images contain faces of people of various age, gender and ethnicity. We use the unaligned cropped faces in our experiments. We randomly divide the data set into a training set of 16,595 images (70%), a validation set of 3,556 images (15%) and a test set of 3,557 images (15%). We consider two tasks on UTKFace: gender recognition (binary classification) and age estimation (regression).

MOROCO (Butnaru & Ionescu, 2019) is a data set of 33,564 news articles that are written in Moldavian or Romanian. Each text sample has an average of 309 tokens. The main task is to discriminate between the Moldavian and the Romanian dialects. The data set is divided into a training set of 21,719 text samples, a validation set of 5,921 text samples and a test set of 5,924 text samples.

4.2. Generic Experimental Setup

Evaluation metrics. For the classification tasks (object class recognition, gender prediction and dialect identification), we employ the classification accuracy as evaluation metric. For the regression task (age estimation), we report the mean absolute error (MAE).

Baselines. We consider several neural architectures ranging from shallow MLPs to deep CNNs: 1-hidden layer MLP, 2-hidden layers MLP, LeNet, VGG-11, ResNet-50 and character-level CNNs with and without Squeeze-and-Excitation blocks (Hu et al., 2018). The baseline architectures are based on ReLU or leaky ReLU. The goal of our experiments is to study the effect of (i) replacing ReLU and leaky ReLU with ADA and leaky ADA, respectively, and (ii) replacing the standard neurons with our PyNADA. All neural models are trained using the Adam optimizer (Kingma & Ba, 2015). With the exception of ResNet-50, which is implemented in PyTorch (Paszke et al., 2019), all other neural networks are implemented in TensorFlow (Abadi et al., 2016). Since we employ different architectures in each task, we describe them in more details in the corresponding subsections below.

Hyperparameter tuning. We tune the basic hyperparameters such as the learning rate, the mini-batch size and the number of epochs, for each neural architecture using

Table 1. Object class recognition accuracy rates (in %) for four neural models (1-hidden layer MLP, 2-hidden layers MLP, LeNet, VGG-11) on Fashion-MNIST. Results are reported with various activations (ReLU, leaky ReLU, ADA, leaky ADA) and artificial neurons (standard and PyNADA). Test results of models that are significantly better than corresponding baselines (ReLU or leaky ReLU), according to a paired McNemar's test (Dietterich, 1998), are marked with † or ‡ for the significance levels 0.05 or 0.01, respectively.

MODEL	ACTIVATION	α	VALIDATION	TEST
1-HIDDEN LAYER MLP	RELU	-	89.77	88.88
1-HIDDEN LAYER MLP	LEAKY RELU	-	89.78	88.40
1-HIDDEN LAYER MLP	ADA	0.3	89.73	88.98
1-HIDDEN LAYER MLP	LEAKY ADA	0.3	89.63	88.97‡
1-HIDDEN LAYER MLP+PYNADA	RELU, ADA	LEARNABLE	90.29	89.45‡
1-HIDDEN LAYER MLP+PYNADA	LEAKY RELU, LEAKY ADA	LEARNABLE	90.45	89.34‡
2-HIDDEN LAYERS MLP	RELU	-	89.77	88.71
2-HIDDEN LAYERS MLP	LEAKY RELU	-	89.52	88.18
2-HIDDEN LAYERS MLP	ADA	LEARNABLE	89.62	88.99
2-HIDDEN LAYERS MLP	LEAKY ADA	0.1	89.98	88.93‡
2-HIDDEN LAYERS MLP+PYNADA	RELU, ADA	LEARNABLE	90.34	89.40‡
2-HIDDEN LAYERS MLP+PYNADA	LEAKY RELU, LEAKY ADA	LEARNABLE	90.13	89.27‡
LENET	RELU	-	91.90	90.84
LENET	LEAKY RELU	-	91.76	90.88
LENET	ADA	0.3	92.05	91.34†
LENET	LEAKY ADA	0.9	91.71	91.38†
LENET+PYNADA	RELU, ADA	0.3	91.78	91.27
LENET+PYNADA	LEAKY RELU, LEAKY ADA	LEARNABLE	92.22	91.60‡
VGG-11	RELU	-	94.07	93.38
VGG-11	LEAKY RELU	-	93.90	93.21
VGG-11	ADA	1.0	94.15	93.48
VGG-11	LEAKY ADA	0.3	94.22	93.66†
VGG-11+PYNADA	RELU, ADA	0.3	94.27	93.54
VGG-11+PYNADA	LEAKY RELU, LEAKY ADA	1.0	94.24	93.47

grid search on the validation set of the corresponding task. We consider learning rates between 10^{-3} and 10^{-5}, mini-batches of 10, 32, 64 or 128 samples and numbers of epochs between 10 and 100. The parameter tuning is performed using the baseline architectures based on ReLU. Once tuned, the same parameters are used for ADA, leaky ADA, PyNADA and leaky PyNADA[2]. Hence, the basic parameters are tuned in favor of ReLU. The leak parameter for leaky ReLU and leaky ADA is set to $l = 0.01$ in all the experiments, without further tuning. We note that (leaky) ADA and (leaky) PyNADA have two additional hyperparameters that require tuning on validation. For the constant c, we consider two possible values, either 0 or 1. For the parameter α, we consider two options: (a) perform grid search in the range $[0.1, 1]$ using a step of 0.1, or (b) learn α using gradient descent during training.

4.3. Results on Fashion-MNIST

Neural architectures. For the Fashion-MNIST data set, we consider two MLPs and two CNNs (LeNet, VGG-11). We kept the same architecture as in the original paper for VGG-11 (Simonyan & Zisserman, 2014), but for LeNet (LeCun et al., 1998), we replaced the average-pooling layer with max-pooling. The first MLP architecture (MLP-1) is composed of one hidden layer with 100 units and one output layer with 10 units (the number of classes). The second MLP has two hidden layers with 100 units and 10 units, respectively, followed by the output layer with another 10 units. The considered MLP architectures are similar to those attaining better results among the MLP architectures evaluated by Xiao et al. (2017).

Hyperparameter tuning. We trained LeNet for 30 epochs, using a learning rate of 10^{-3} for the first 15 epochs and 10^{-4} for the last 15 epochs. We trained MLP-1 and MLP-2 in the same manner as LeNet. However, VGG-11 was trained for 100 epochs, starting with a learning rate of 10^{-4} in the first 50 epochs, decreasing it to 10^{-5} in the last 50 epochs. We trained all models on mini-batches of 64 images. In all the experiments with (leaky) ADA or (leaky) PyNADA, we either validated or learned α and we set the parameter $c = 0$.

Results. We present the results obtained on Fashion-MNIST in Table 1. Our baseline MLP architectures obtain better

[2]Note that for PyNADA we consider both standard and leaky activations. In leaky PyNADA, the basal dendrites are followed by leaky ReLU and the apical dendrites are followed by leaky ADA.

Table 2. Gender prediction accuracy rates (in %) and age estimation MAEs for ResNet-50 on UTKFace. Results are reported with various activations (ReLU, leaky ReLU, ADA, leaky ADA) and artificial neurons (standard and PyNADA). Test results of models that are significantly better than corresponding baselines (ReLU or leaky ReLU), according to a paired McNemar's test (Dietterich, 1998), are marked with ‡ for the significance level 0.01.

TASK	MODEL	ACTIVATION	α	VALIDATION	TEST
GENDER PREDICTION	RESNET-50	RELU	-	90.11	88.56
	RESNET-50	LEAKY RELU	-	89.93	88.91
	RESNET-50	ADA	0.1	90.06	88.29
	RESNET-50	LEAKY ADA	0.1	89.86	89.12
	RESNET-50+PYNADA	RELU, ADA	0.5	90.36	90.66‡
	RESNET-50+PYNADA	LEAKY RELU, LEAKY ADA	0.5	90.51	90.80‡
AGE ESTIMATION	RESNET-50	RELU	-	6.16	6.39
	RESNET-50	LEAKY RELU	-	5.85	6.01
	RESNET-50	ADA	LEARNABLE	5.66	5.91‡
	RESNET-50	LEAKY ADA	LEARNABLE	5.68	5.88‡
	RESNET-50+PYNADA	RELU, ADA	0.5	5.44	5.79‡
	RESNET-50+PYNADA	LEAKY RELU, LEAKY ADA	0.5	5.60	5.83‡

accuracy rates than those of Xiao et al. (2017), e.g. the difference obtained for MLP-1 with ReLU is 1.78% (we report 88.88%, while Xiao et al. (2017) report 87.10%). We notice that leaky ReLU obtains slightly lower accuracy rates than ReLU, the only exception being the result of LeNet on the test set. When we replace ReLU with ADA, we notice a slight accuracy drop on the validation set for the MLP architectures, but, for the CNN architectures, the validation accuracy improves. Nonetheless, for both MLP and CNN architectures, the accuracy of ADA on the test set improves by up to 0.5% over ReLU. We obtain a significant improvement, from 90.84% to 91.34%, for the replacement of ReLU with ADA using LetNet. Interestingly, we notice that the value of the cross-entropy loss is always lower (on both validation and test sets) when we use ADA instead of ReLU. When we employ PyNADA, the accuracy rates on the validation and the test sets improve for all the architectures. The largest improvement of PyNADA on the test set is 0.69%, obtained with MLP-2. Our highest absolute gain on Fashion-MNIST is 1.09%, obtained using MLP-2 with leaky PyNADA (89.27%) instead of leaky ReLU (88.18%).

4.4. Results on UTKFace

Neural architecture. For gender prediction and age estimation, we employ the ResNet-50 architecture (He et al., 2016). Residual networks use skip connections to propagate information over convolutional layers, avoiding the vanishing gradient problem. This enables effective training of very deep models, e.g. ResNet-50 is formed of 50 layers.

Hyperparameter tuning. All ResNet-50 variants are trained on mini-batches of 10 samples using a learning rate equal to 10^{-4}. The models are trained for 15 epochs on the gender prediction task, and for 100 epochs on the age estimation task. For (leaky) ADA and (leaky) PyNADA, we either validate or tune the parameter α, in the same time setting the parameter $c = 0$.

Results. We present the gender prediction and age estimation results in Table 2. In the gender prediction task, we notice that ADA yields slightly lower results than ReLU, while leaky ADA attains slightly better results than leaky ReLU on the test set. We observe more significant differences in favor (leaky) PyNADA versus (leaky) ReLU. Our highest absolute gain (2.1%) in the gender prediction task is obtained when ResNet-50 is equipped with PyNADA (90.66%) instead of standard neurons with ReLU (88.56%). In the age estimation task, we notice that all versions of (leaky) ADA and (leaky) PyNADA surpass (leaky) ReLU by significant margins. With an MAE of 5.44 on the validation set and an MAE of 5.79 on the test set, PyNADA attains the best results in age estimation. With respect to ReLU, PyNADA reduces the average error by 0.72 years on validation and by 0.6 years on test.

4.5. Results on MOROCO

Neural architectures. For dialect identification, we consider the character-level CNN models presented in (Butnaru & Ionescu, 2019), which follow closely the model of Zhang et al. (2015). The two CNNs share the same architecture, being composed of an embedding layer, followed by three convolutional and max-pooling blocks, two fully-connected layers with dropout 0.5 and the final softmax classification layer. The second architecture contains attention in the form of Squeeze-and-Excitation (SE) blocks (Hu et al., 2018) inserted after every convolutional layer. For the SE blocks, we set the reduction ratio to 64. For both architectures, we keep the same size for the embedding (256) and the same number of convolutional filters (128) as Butnaru and Ionescu (2019). In fact, our baseline architectures (CNN and CNN+SE) are

Table 3. Dialect identification accuracy rates (in %) and training times (in seconds per epoch) for two character-level neural models (CNN and CNN+SE) on MOROCO. Results are reported with various activations (ReLU, leaky ReLU, ADA, leaky ADA) and artificial neurons (standard and PyNADA). Training times are measured on a computer with Nvidia GeForce GTX 1080 GPU with 11GB of RAM. Test results of models that are significantly better than corresponding baselines (ReLU or leaky ReLU), according to a paired McNemar's test (Dietterich, 1998), are marked with † or ‡ for the significance levels 0.05 or 0.01, respectively.

MODEL	ACTIVATION	α	VALIDATION	TEST	TIME(S)
CNN	RELU	-	92.99	92.79	27
CNN	LEAKY RELU	-	93.04	92.90	30
CNN	ADA	LEARNABLE	93.97	93.68†	33
CNN	LEAKY ADA	0.4	93.67	93.55†	37
CNN+PYNADA	RELU, ADA	LEARNABLE	93.95	93.61†	52
CNN+PYNADA	LEAKY RELU, LEAKY ADA	LEARNABLE	93.81	93.53†	57
CNN+SE	RELU	-	93.02	92.99	42
CNN+SE	LEAKY RELU	-	93.08	93.06	45
CNN+SE	ADA	LEARNABLE	93.83	93.99‡	47
CNN+SE	LEAKY ADA	LEARNABLE	93.70	93.49†	51
CNN+SE+PYNADA	RELU, ADA	0.5	93.81	93.72†	70
CNN+SE+PYNADA	LEAKY RELU, LEAKY ADA	LEARNABLE	93.90	93.61†	75

completely identical to those of Butnaru and Ionescu (2019).

Hyperparameter tuning. Since Butnaru and Ionescu (2019) already tuned the hyperparameters of the character-level CNNs on the MOROCO validation set, we decided to use the same parameters and skip the grid search. Hence, we set the learning rate to $5 \cdot 10^{-4}$ and use mini-batches of 128 samples. Each CNN is trained for 50 epochs, keeping the model with highest validation accuracy for evaluation on test. For (leaky) ADA and (leaky) PyNADA, we obtain optimal results with $c = 1$, while α is either validated or optimized during training.

Results. We present the dialect identification results on MOROCO in Table 3. First, we observe the baseline CNN and CNN+SE models confirm the results reported by Butnaru and Ionescu (2019). We also note that the reported the accuracy rates are consistent across validation and test, for all models. For the character-level CNN (without SE blocks), we obtain the largest improvement on the test set when we replace ReLU (92.79%) with ADA (93.68%). Nonetheless, the improvements of leaky ADA, PyNADA and leaky PyNADA are all higher than 0.6% on the test set, and the differences are statistically significant. The results are somewhat consistent among the two architectures, CNN and CNN+SE. For example, for the CNN+SE model, we report the largest improvement on the test set by replacing ReLU (92.99%) with ADA (93.99%), just as for the CNN without SE blocks. Our highest absolute gain on MOROCO is 1%. Overall, the results indicate that all variants of (leaky) ADA and (leaky) PyNADA attain significantly better results than (leaky) ReLU.

Running time. Since ADA needs to compute the exponential function, it is more computational intensive than ReLU. Moreover, PyNADA has twice more weights than a standard neuron. Hence, in addition to the accuracy rates, we report the training time (in seconds per epoch) in Table 3. With respect to (leaky) ReLU, it seems that (leaky) ADA requires 5 or 6 additional seconds per epoch, which means that the training times increases by 15% or 20%. Meanwhile, (leaky) PyNADA seems to need about 25 to 30 extra seconds compared to (leaky) ReLU, increasing the training time by 70% or 100%. We thus conclude that the accuracy improvements brought by (leaky) ADA and (leaky) PyNADA come with a non-negligible computational cost.

5. Conclusion

In this paper, we proposed a biologically-inspired activation function and a new model of artificial neuron. The novel apical dendrite activation function (i) enables individual artificial neurons to solve non-linearly separable problems such as the XOR logical function and (ii) brings significant performance improvements for a broad range of neural architectures and tasks. Our research also opens a few directions for future research. Since the activation damps along the positive side of the domain, we believe it is worth investigating if ADA is more robust to out-of-distribution or adversarial examples. As the gradient saturates on the positive side, other directions of study are to inject noise into ADA to avoid saturation (Gulcehre et al., 2016) or to employ alternative optimization methods (that do not rely on gradients) in conjunction with ADA. In future work, we could also study if ADA or PyNADA are more useful in certain layers.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: A system for large-scale machine learning. In *Proceedings of OSDI*, pp. 265–283, 2016.

Butnaru, A. M. and Ionescu, R. T. MOROCO: The Moldavian and Romanian Dialectal Corpus. In *Proceedings of ACL*, pp. 688–698, 2019.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *Proceedings of ICLR*, 2016.

Dietterich, T. G. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

Gidon, A., Zolnik, T. A., Fidzinski, P., Bolduan, F., Papoutsi, A., Poirazi, P., Holtkamp, M., Vida, I., and Larkum, M. E. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87, 2020.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout Networks. In *Proceedings of ICML*, volume 28, pp. 1319–1327, 2013.

Guerguiev, J., Lillicrap, T. P., and Richards, B. A. Towards deep learning with segregated dendrites. *eLife*, 6:e22901, 2017.

Gulcehre, C., Moczulski, M., Denil, M., and Bengio, Y. Noisy Activation Functions. In *Proceedings of ICML*, pp. 3059–3068, 2016.

Hawkins, J. and Ahmad, S. Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Frontiers in Neural Circuits*, 10:23, 2016.

Hayou, S., Doucet, A., and Rousseau, J. On the Impact of the Activation function on Deep Neural Networks Training. In *Proceedings of ICML*, pp. 2672–2680, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of ICCV*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of CVPR*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

Hu, J., Shen, L., and Sun, G. Squeeze-and-Excitation Networks. In *Proceedings of CVPR*, pp. 7132–7141, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

Knutson, B. and Cooper, J. C. The Lure of the Unknown. *Neuron*, 51(3):280–282, 2006.

Kohonen, T. The Self-Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference Target Propagation. In *Proceedings of ECML/PKDD*, pp. 498–515, 2015.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of WDLASL*, 2013.

Minsky, M. and Papert, S. A. *Perceptrons: An introduction to computational geometry*. MIT press, 1969.

Nair, V. and Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of ICML*, pp. 807–814, 2010.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of NeurIPS*, pp. 8024–8035, 2019.

Rosenblatt, F. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of ICLR*, 2014.

Wang, X., Gao, L., Song, J., Zhen, X., Sebe, N., and Shen, H. T. Deep appearance and motion learning for egocentric activity recognition. *Neurocomputing*, 275:438–447, 2018.

Widrow, B. An Adaptive 'Adaline' Neuron Using Chemical 'Memistors'. Technical Report 1553-2, Stanford Electronics Laboratories, 1960.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xu, D., Yan, Y., Ricci, E., and Sebe, N. Detecting anomalous events in videos by learning deep representations of appearance and motion. *Computer Vision and Image Understanding*, 156:117–127, 2017.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *Proceedings of NIPS*, pp. 649–657, 2015.

Zhang, Z., Song, Y., and Qi, H. Age Progression/Regression by Conditional Adversarial Autoencoder. In *Proceedings of CVPR*, pp. 5810–5818, 2017.