Deeper Bidirectional Neural Networks with Generalized Non-Vanishing Hidden Neurons

Olaoluwa Adigun
Signal and Image Processing Institute
Department of Electrical and Computer Engineering
Los Angeles, California 90089-2564.
adigun@usc.edu

Bart Kosko
Signal and Image Processing Institute
Department of Electrical and Computer Engineering
Los Angeles, California 90089-2564.
kosko@usc.edu

***Abstract*—The new NoVa hidden neurons have outperformed ReLU hidden neurons in deep classifiers on some large image test sets. The NoVa or nonvanishing logistic neuron additively perturbs the sigmoidal activation function so that its derivative is not zero. This helps avoid or delay the problem of vanishing gradients. We here extend the NoVa to the generalized perturbed logistic neuron and compare it to ReLU and several other hidden neurons on large image test sets that include CIFAR-100 and Caltech-256. Generalized NoVa classifiers allow deeper networks with better classification on the large datasets. This deep benefit holds for ordinary unidirectional backpropagation. It also holds for the more efficient bidirectional backpropagation that trains in both the forward and backward directions.**

***Index Terms*—Bidirectional backpropagation, Logistic, ReLU, NoVa hidden neurons, vanishing gradient.**

I. Better Hidden Activations for Deeper Learning

We propose the new *generalized nonvanishing* or G-NoVa activation for hidden neurons in deep neural classifiers. Simulations show that it outperforms its main hidden-activation rivals on very deep multilayer perceptron neural classifiers on three image datasets. The comparative benefits of the G-NoVa neurons were most pronounced for the deepest classifiers.

Figures 1 and 2 show the graphs of the activation contenders and their first derivatives: logistic, linear (identity), ReLU, leaky ReLU, Swish, NoVa, and G-NoVa activations. The next section gives their mathematical definitions.

The G-NoVa neuron is a type of *perturbed* logistic activation. The recent nonvanishing or NoVa neuron is an additively perturbed logistic whose derivative does not vanish [1]. So the NoVa neuron avoids the vanishing gradient problem that so quickly overtakes logistic hidden neurons in deep neural networks.

The new G-NoVa neuron is more general than the NoVa neuron because it combines both an additive and a multiplicative perturbation with the traditional logistic sigmoid. This leads to superior classifier performance on the large image datasets that we tested. It also suggests that the decades-old logistic model neuron may be more biologically plausible because any such real neuron would involve noise perturbations that are both additive and multiplicative [2], [3].

We specifically found that the G-NoVa outperformed the popular ReLU or rectified linear unit hidden activation and its leaky variant. The ReLU neuron has become the default hidden neuron for modern deep neural classifiers [4]. It is a threshold linear neuron and dates back to at least Fukushima's experiments with multilayer networks or neocognitrons [5], [6]

We further found that G-NoVa deep networks outperformed its activation rivals for the more general case of training with *bidirectional* backpropagation [7], [8]. Figure 3 shows the forward and backward probability flow in bidirectional backpropagation (B-BP).

Ordinary backpropagation is unidirectional despite its name. It ignores the rich associative and probabilistic information that the multilayer network contains in its backward direction as it trains on input-output associations. That is why running an ordinary BP-trained network backwards from unit-bit-vector class labels to input pixels or data registers produces only visual noise at the input layer. Bidirectional BP or B-BP tends to produce a centroidal estimate at the input layer of what the network *expects* to see at the input given the current input stimulation and given the input-output associations that the network's web of synapses has learned. B-BP incurs trivial extra cost for training and yet fully exploits the probabilistic information in the joint forward and backward likelihoods (or joint posteriors) in the layered network. It resembles Grossberg's earlier and unsupervised ART or adaptive resonance theory [9], [10] in the bidirectional sense that the network's backward projections endow the network with a type of attentive focus or expectation at the input layer [11], [12].

Figure 3 also reveals the hidden regressor in the backward direction of every deep neural classifier. The network's forward direction maps an input pattern vector $\mathbf{x}$ to probability vector rounded off to a unit bit vector at the output $N(\mathbf{x})$. The K output softmax neurons of the classifier give a forward-pass likelihood $p(\mathbf{y}|\mathbf{x}, \Theta)$ as a one-shot multinomial probability or a single roll of an unfair K-sided die. So the forward pass seeks to minimize the negative log-likelihood or the cross-entropy. But the backward pass through the transposes of the weight matrices gives a value $N^T(\mathbf{y})$ at the input layer of identity neurons or data registers. These identity activations give the backward likelihood $p(\mathbf{x}|\mathbf{y}, \Theta)$ as a vector normal (or vector Laplacian) and thus the network seeks the input sample centroid as it minimizes the input squared error at the input layer. B-BP maximizes the *joint* likelihood $p(\mathbf{y}|\mathbf{x}, \Theta)p(\mathbf{x}|\mathbf{y}, \Theta)$ and thus the joint log-likelihood $\log p(\mathbf{y}|\mathbf{x}, \Theta) + \log p(\mathbf{x}|\mathbf{y}, \Theta)$ and

so minimizes the joint error of cross-entropy in the forward direction and squared error in the backward direction.

The G-NoVa hidden networks still outperformed their activation rivals when trained with B-BP. They also tended to perform better in the bidirectional case than in the unidirectional case. The classification results in Tables II - VII show that the G-NoVa hidden neurons did best for both BP and B-BP with more pronounced benefits for the larger image training sets and deeper network architectures. Figures 4 - 6 show image samples from the respective CIFAR-10, CIFAR-100, and Caltech-256 image datasets. The singly perturbed NoVa hidden networks were the runner-up to the G-NoVa networks. The simple linear or identity activation $a(x) = x$ also did surprisingly well in these deeper network on the larger image datasets. Figure 7 shows these results for unidirectional BP training plotted against the number of hidden layers. Figure 7 shows the results for B-BP also plotted against the number of hidden layers.

These classification results on the CIFAR-10, CIFAR-100, and Caltech-256 image datasets suggest that neural engineers should consider experimenting both with the new G-NoVa hidden neuron and with the more general B-BP bidirectional paradigm of supervised training. The experiments ran with multilayer perceptron (MLP) classifiers.

II. Old and New Activation Functions

This section reviews the main hidden activations in use as well as the new NoVa and the even newer generalized or G-NoVa activations. Simulations compared deep networks using these activations on the image datasets CIFAR-10, CIFAR-100, and Caltech-256 for both ordinary unidirectional backpropagaiton and for the more general bidirectional backpropagation training algorithm.

The term a_j^h denotes the activation of the j^{th} neuron in layer h and o_j^h is the input to the neuron. The term $a_j^{h\prime}$ denotes the corresponding derivative.

A. Logistic Sigmoid

The sigmoidal logistic activation acts as a smooth or soft threshold. So it has served as a model neuron for decades [13], [14]. The logistic endows a neural network with proven nonlinearity approximation power [15] and has a simple closed-form derivative. But it suffers from the problem of vanishing gradient [16] precisely because of the form of its derivative.

The logistic activation function involves a ratio of exponentials. It describes two-hypothesis Bayesian classification as in simple logistic regression (compared with the more general softmax output neuron that describes multi-class Bayesian classification or so-called multinomial regression) [17]. The logistic has the ratio form

$$a_j^h = \sigma(c\ o_j^h) = \frac{1}{1 + \exp^{-c\ o_j^h}} \tag{1}$$

with derivative

$$a_j^{h\prime} = \frac{da_j^h}{do_j^h} = c\ \sigma(c\ o_j^h)(1 - \sigma(c\ o_j^h)) \tag{2}$$

$$= \frac{c\ \exp^{-c\ o_j^h}}{\left(1 + \exp^{-c\ o_j^h}\right)^2}. \tag{3}$$

So the derivative vanishes for extreme values or their machine-word equivalents: $a_j^{h\prime} = 0$ if $\sigma = 0$ or $\sigma = 1$.

Figures 1a and 2a show the respective activation and derivative for a logistic sigmoid neuron. The logistic activation is smooth everywhere and indeed is a diffeomorphism. But in practice it "dies" for extreme input values.

B. Leaky and Ordinary Rectified Linear Units

The leaky ReLU or LReLU activation modifies the threshold-linear structure of the ordinary ReLU activation. It uses the identity function $a(x) = x$ on the positive domain and scales the negative domain by $c \geq 0$:

$$a_j^h = LReLU(o_j^h) = \begin{cases} c\ o_j^h, & o_j^h \leq 0 \\ o_j^h, & o_j^h > 0 \end{cases} \tag{4}$$

with derivative

$$a_j^{h\prime} = \frac{da_j^h}{do_j^h} = \begin{cases} c, & o_j^h < 0 \\ 1, & o_j^h > 0 \end{cases}. \tag{5}$$

The leaky ReLU uses $c > 0$ [18].

The leaky ReLU's nonlinear approximation power is low because this function is the identity function over the positive domain and is scaled linear over its negative domain. The derivative of the leaky ReLU is not defined at $o_j^h = 0$. Figures 1b and 2b show the respective activation function and derivative of a leaky ReLU neuron.

Setting $c = 0$ gives the non-leaky or ordinary rectified linear unit or ReLU. This threshold-linear unit truncates or rectifies the negative domain by setting the function equal to zero there [19]–[21]. The derivative of a ReLU activation equals zero over the negative domain. Deep neural networks with hidden ReLU neurons suffer from dying neurons [20], [22], [23]. Figures 1c and 2c show the respective activation function and derivative of a non-leaky ReLU neuron. ReLU activation applies to tasks in speech recognition, computer vision, and other areas [18].

C. Swish

The Swish activation is a scaled logistic [24]:

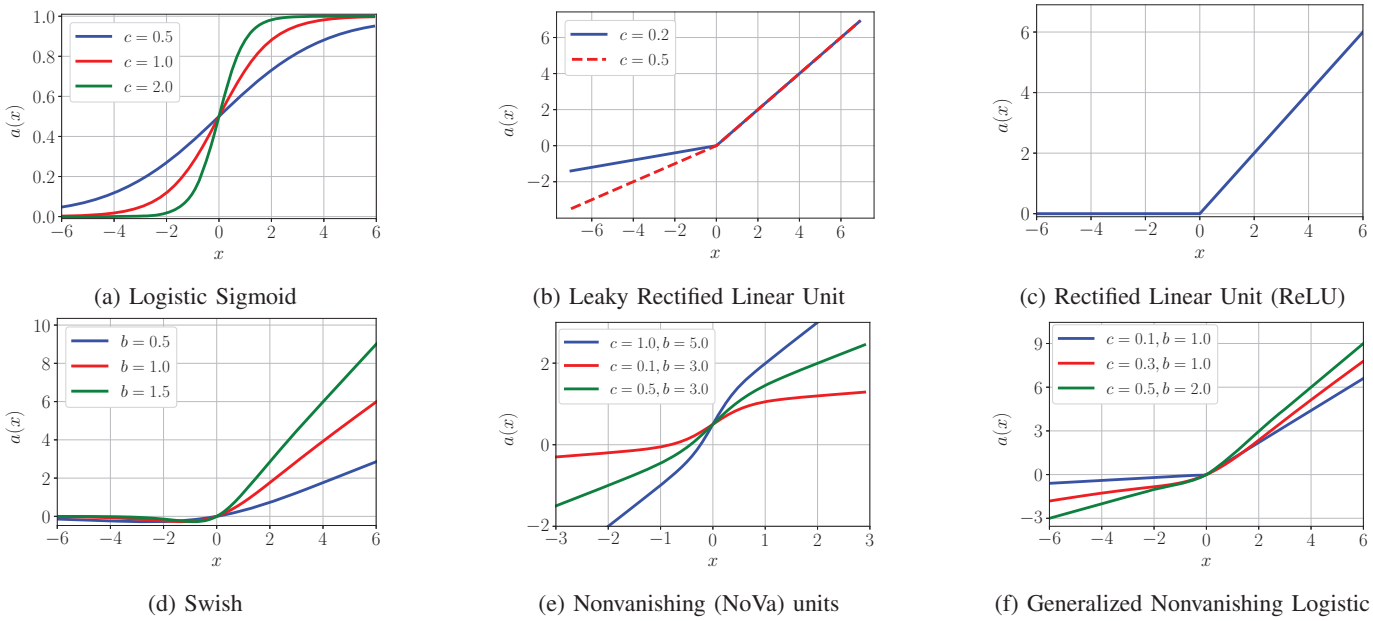
$$a_j^h = o_j^h \sigma(b\ o_j^h) = \frac{o_j^h}{1 + \exp^{-(b\ o_j^h)}} \tag{6}$$

and the corresponding derivative is

$$a_j^{h\prime} = \sigma(b\ o_j^h)\big(1 + (b\ o_j^h)(1 - \sigma(b\ o_j^h))\big) \tag{7}$$

$$= \frac{1 + \exp^{-(b\ o_j^h)}\left(1 + (b\ o_j^h)\right)}{\left(1 + \exp^{-(b\ o_j^h)}\right)^2}. \tag{8}$$

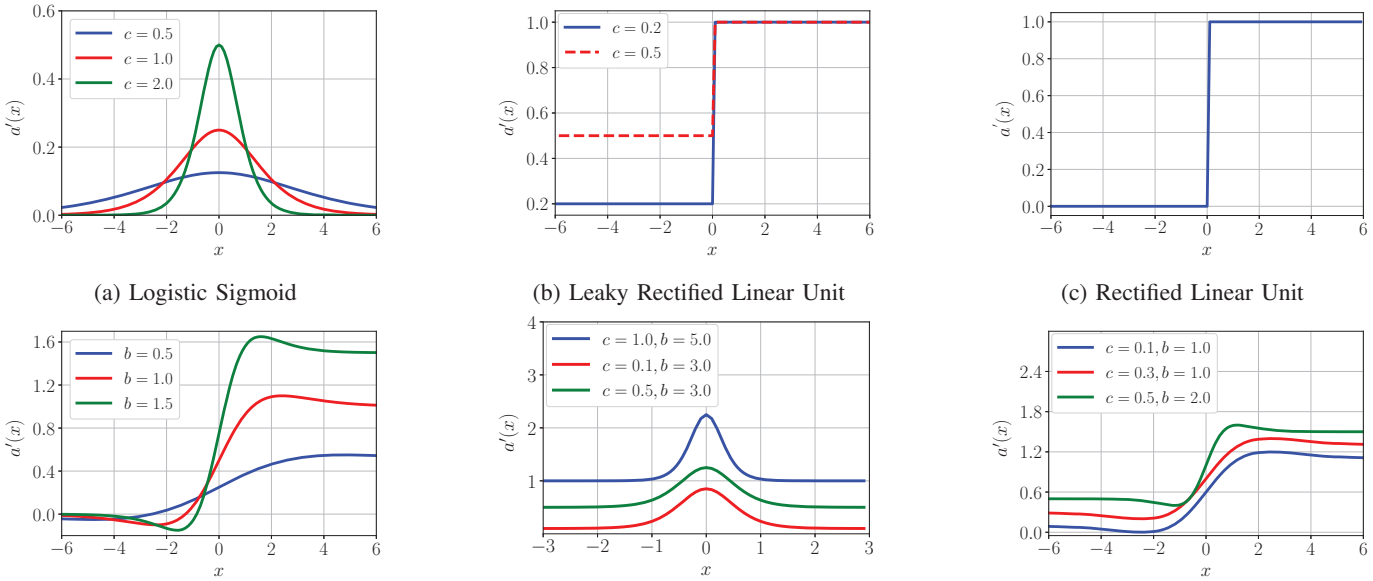
Figures 1d and 2d show the respective activation and derivative of a swish neuron.



(a) Logistic Sigmoid (b) Leaky Rectified Linear Unit (c) Rectified Linear Unit (ReLU)

(d) Swish (e) Nonvanishing (NoVa) units (f) Generalized Nonvanishing Logistic

Fig. 1: Activation functions for multilayer neural networks: logistic sigmoid, leaky rectified linear unit (LReLU), threshold linear or non-leaky ReLU, Swish, NoVa unit, and the new generalized nonvanishing (G-NoVa) unit.



(a) Logistic Sigmoid (b) Leaky Rectified Linear Unit (c) Rectified Linear Unit

(d) Swish (e) Nonvanishing Logistic (f) Generalized Nonvanishing Logistic

Fig. 2: Derivatives of neural activation functions : logistic sigmoid, leaky rectified linear unit (LReLU), threshold linear or non-leaky ReLU, swish, Nonvanishing (NoVa) logistic, and the new generalized nonvanishing (G-NoVa) logistic neuron.

D. Nonvanishing (NoVa) Unit

Th recent NoVa activation function is an additively perturbed logistic sigmoid [1] with derivative that is never zero:

$$a_j^h = c\ o_j^h + \sigma(b\ o_j^h) = \frac{1 + c\ o_j^h(1 + \exp^{-b\ (o_j^h)})}{1 + \exp^{-(b\ o_j^h)}} \tag{9}$$

with derivative

$$a_j^{h\prime} = \frac{da_j^h}{do_j^h} = c + \sigma(b\ o_j^h)\big(1 - \sigma(b\ o_j^h)\big) \tag{10}$$

$$= c + \frac{b\ \exp^{-(b\ o_j^h)}}{\big(1 + \exp^{-(b\ o_j^h)}\big)^2} \tag{11}$$

where $c \geq 0$ and $b \geq 0$. Figures 1e and 2e show the respective activation and derivative of a NoVa neuron.

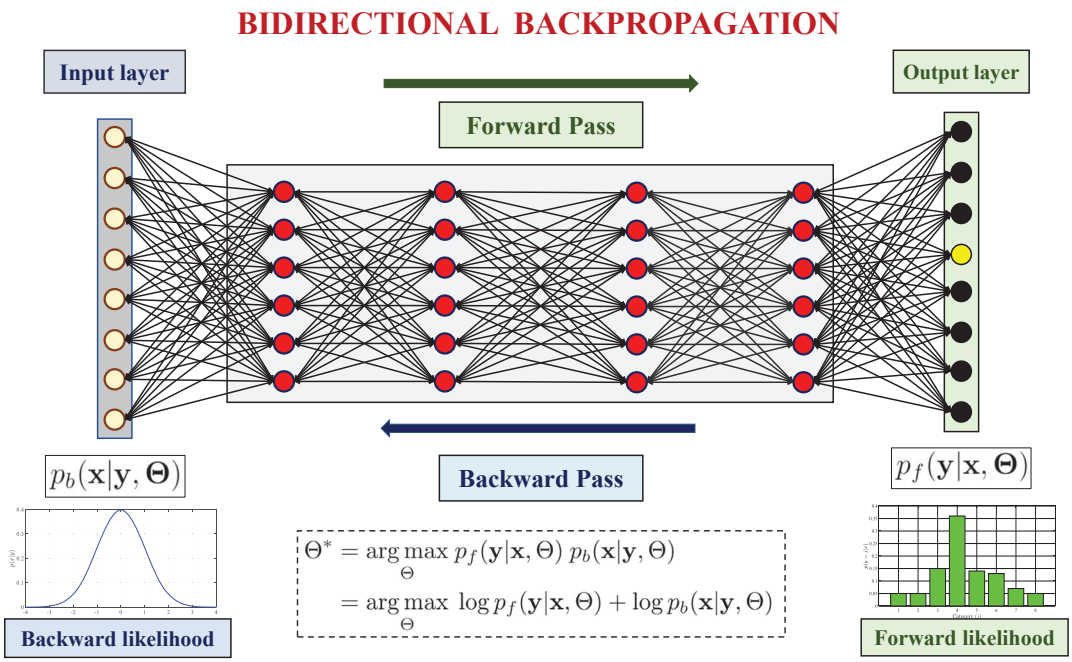


Fig. 3: Bidirectional Backpropagation learning. B-BP maximizes the *joint* log-likelihood of the forward network likelihood $p_f(\mathbf{y}|\mathbf{x},\Theta)$ and the backward likelihood $p_b(\mathbf{x}|\mathbf{y},\Theta)$. The forward likelihood is a one-shot multinomial since the output neurons have softmax activations. The backward layer in a classifier is vector normal since the input neurons have linear or identity activations and thus B-BP reveals a *hidden regressor* in the backward direction. B-BP equivalently minimizes the negative of the summed log-likelihoods. So it minimizes the sum of a cross-entropy and a squared error. Ordinary unidirectional backpropagation minimizes only the forward cross-entropy.

E. Generalized Nonvanishing (G-NoVa) Unit

The new G-NoVa activation function is an additively *and* multiiplicatively perturbed logistic sigmoid:

$$a_j^h = c\ o_j^h + o_j^h \sigma(b\ o_j^h) \tag{12}$$

$$= \frac{c\ o_j^h (1 + o_j^h + \exp^{-(b\ o_j^h)})}{1 + \exp^{-(b\ o_j^h)}} \tag{13}$$

with derivative

$$a_j^{h\prime} = \frac{da_j^h}{do_j^h} = c + \sigma(b\ o_j^h)\big(1 + b\ o_j^h(1 - \sigma(b\ o_j^h))\big) \tag{14}$$

$$= c + \frac{1 + \exp^{-(b\ o_j^h)}\ \big(1 + (b\ o_j^h)\big)}{\big(1 + \exp^{-(b\ o_j^h)}\big)^2} \tag{15}$$

with $c \geq 0$ and $b \geq 0$. G-NoVa simplifies to swish function with $c = 0$ and $b > 0$. It simplifies to a linear function with $c \geq 0$ and $b = 0$.

The vanishing gradient problem results if the value of all or most of the neuronal derivatives fall within the range of $(-1, 1)$ but G-NoVa is not susceptible to this. This is so because the G-NoVa parameters c and b control the derivatives outside the fractional range of $(-1, 1)$. This allows the product of multiple derivatives (from the chain rule of BP algorithm) not to tend towards zero and consequently avoid vanishing gradient. Figure 2f shows the the derivative of some G-NoVa units and for *most* of the positive input ($x \in \mathbb{R}^+$) the derivative $a'(x) \geq 1$. This reduces the tendency of the derivative of deep networks to tend towards zero.

III. Bidirectional Backpropagation

Bidirectional backpropagation (B-BP) extends unidirectional BP by training both the forward and backward flow of neural signals through a multilayer deep neural network [7], [25].

Figure 3 shows that B-BP maximizes the network's joint log-likelihood (or log-posterior) $\log p(\mathbf{y}|\mathbf{x},\Theta) + \log p(\mathbf{x}|\mathbf{y},\Theta)$ for network parameters Θ. So it equivalently minimizes the sum of the joint errors. These are the cross-entropy in the forward direction since $p(\mathbf{y}|\mathbf{x},\Theta)$ is a one-shot multinomial in the forward direction and the squared-error in the backward direction since a vector normal probability describes the identity neurons at the input layer. Thus B-BP reveals a *hidden regressor* in the backward direction of a classifier. This allows the B-BP trained network to run backward and produce a centroidal estimate at the input layer given the current input pattern and given the input-output associations that the network has learned. Running an ordinary BP-trained network backward produces only visual noise at the input. B-BP exploits this backward information at little extra computational cost.

The backward pass uses the transpose matrices of the weight matrices that the forward pass uses. The backward pass $\mathcal{N}^{-1}(\mathbf{y})$ of output $\mathbf{y} \in \mathcal{Y}$ propagates the sample y from the output layer to the input layer through the transpose matrices

that house the weights of the hidden layers [26]. $\mathcal{N}^{-1}(\mathbf{y})$ approximates the inverse mapping $f^{-1}(\mathbf{y}) = \mathbf{x}$ if it exists kosko2021bidirectional. The backward pass $\mathcal{N}^T$ through the transposed weight matrices tends to map an output codeword to the centroid of the inverse-image set $\{\mathbf{x} : f(\mathbf{x}) = \mathbf{y}\}$ [7]. This backward tug toward the inverse centroid acts as a type of trained attentive focus inherent in the network's training from input-output associations [26].

B-BP training seeks to jointly maximize the forward likelihood $p(\mathbf{y}|\mathbf{x}, \Theta)$ and backward likelihood $p(\mathbf{x}|\mathbf{y}, \Theta)$. This training algorithm finds the best weights Θ^* such that

$$\Theta^* = \arg\max_{\Theta} \; p(\mathbf{y}|\mathbf{x}, \Theta) \; p(\mathbf{x}|\mathbf{y}, \Theta). \tag{16}$$

Equation (16) simplifies to maximizing the sum of log-likelihoods because the logarithm is a monotone increasing function. So we can rewrite B-BP as maximizing the sum of the log-likelihoods:

$$\Theta^* = \arg\max_{\Theta} \; \log \, p(\mathbf{y}|\mathbf{x}, \Theta) + \; \log \, p(\mathbf{x}|\mathbf{y}, \Theta) \; . \tag{17}$$

Then Θ^* lies between the maxima of the forward and backward log-likelihoods since the logarithm is strictly concave [26]. The negative log-likelihood equals the error function for a classification network or a regression network [4], [17], [27], [28]. So we can restate the goal of B-BP as solving the minimization problem

$$\Theta^* = \arg\min_{\Theta} \; E_f + \; E_b \tag{18}$$

because $\log \, p(\mathbf{y}|\mathbf{x}, \Theta) = -E_f$ and because $\log \, p(\mathbf{x}|\mathbf{y}, \Theta) = -E_b$. The forward error E_f measures the approximation error between the output vector y and $\mathcal{N}(\mathbf{x})$ for a given input-output pair $(\mathbf{x}, \mathbf{y})$. Classifiers use cross entropy in the case softmax output activation [17], [29] or double cross entropy in the case of logistic output activation [30], [31]. The backward error E_b measures the approximation error between input vector $\mathbf{x}$ and its backward inference $\mathcal{N}^T(\mathbf{y})$ for a given input-output pair $(\mathbf{x}, \mathbf{y})$. It is again squared-error or absolute error for a regression mapping. It can be double cross-entropy at the input layer for a threshold network with steep logistics at the input layer instead of identity neurons [12].

B-BP also extends to Bayesian B-BP for maximizing the joint log-posterior [8]. Then the user specifies a prior for any of the network parameters such as the first set of weights that map the input neurons to the first hidden layer. This paper used only joint-likelihood B-BP and so it impliedly used uniform priors.

TABLE I: Experimental Dataset

Dataset	Training Set	Testing Set	Number of Classes
CIFAR-10	50,000	10,000	10
CIFAR-100	50,000	10,000	100
Caltech-256	23,824	5,956	256

IV. Simulation Results

A. Datasets

The simulated deep classifiers used three image datasets. The first was the CIFAR-10 dataset and the second dataset is the CIFAR-100. The third was the Caltech-256 image dataset. Table I shows the sample distributions of these image datasets

1) CIFAR-10: The CIFAR-10 test set consists of 60,000 color images from 10 categories ($K = 10$). Each image has size $32 \times 32 \times 3$. The 10 pattern categories are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck [32]. Each class consists of 5,000 training samples and 1,000 testing samples. Figure 4 shows sample images with one image per class.

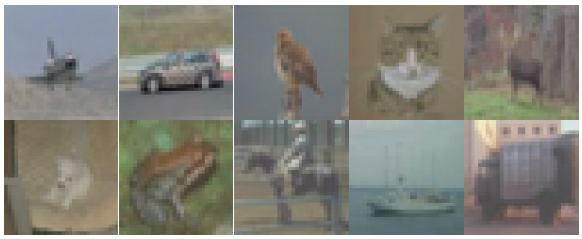


Fig. 4: CIFAR-10 sample images: The figure shows 10 samples from the CIFAR-10 dataset that contains 10 pattern classes and a total of 60,000 sample images.

2) CIFAR-100: CIFAR-100 dataset is a set of 60,000 color images with image size $32 \times 32 \times 3$. The images are from 100 pattern classes with 600 images per class. This extends the CIFAR-10 dataset. Each of the 10 categories of CIFAR-10 further divides into 10 classes. Each class is made up of 500 training images and 100 testing images. Figure 5 shows sample images with one image per class.

3) Caltech-256: This dataset has 30,607 images from 256 pattern classes. Each class has 80 or more images. The 256 classes consist of the two superclasses $animate$ and $inanimate$. The animate superclass contains 69 pattern classes. The inanimate superclass contains 187 pattern classes [33].

We removed the $cluttered$ images and reduced the size of the dataset to 29,780 images. We split the dataset into 23,824 training images and 5,956 test images. The images had different dimensions. We resized each image to $100 \times 100 \times 3$. Figure 6 shows sample images with one image per class.

B. Network Description and Training Parameters

The deep neural classifiers trained on CIFAR-10, CIFAR-100, and Caltech-256 with ordinary and bidirectional backpropagation.

Each classifier network used 500 neurons per hidden layer and used softmax output activations. We varied the size of the hidden layers and the type of hidden activation. The depth or number of hidden layers varied as the values in $\{1, 3, 5, 7, ..., 21\}$. The competing hidden activations were logistic, linear, ReLU, leaky ReLU, Swish, NoVa unit, and G-NoVa unit.

Fig. 5: CIFAR-100 sample images: This figure shows 100 samples from the CIFAR-100 dataset that contains 100 pattern classes with 600 images per class. CIFAR-100 consists of 20 super-classes with 5 classes per super-class.

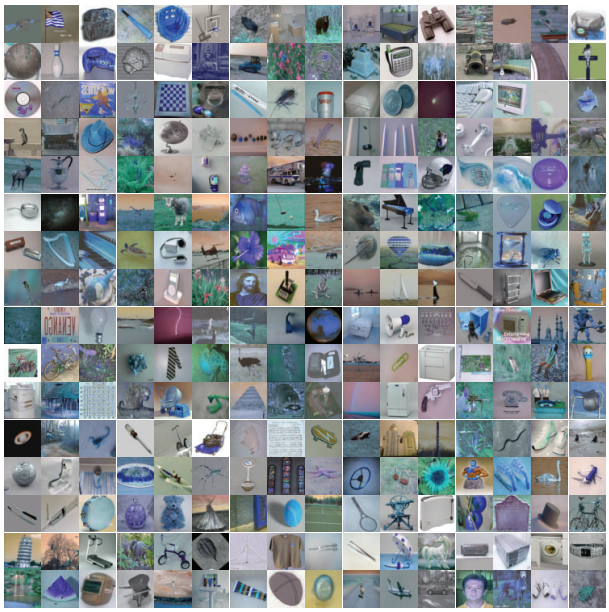
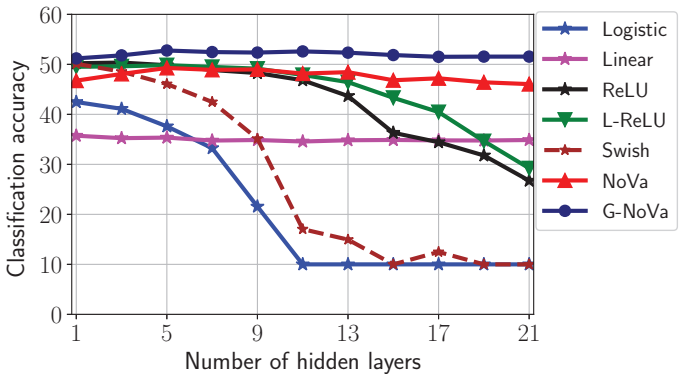
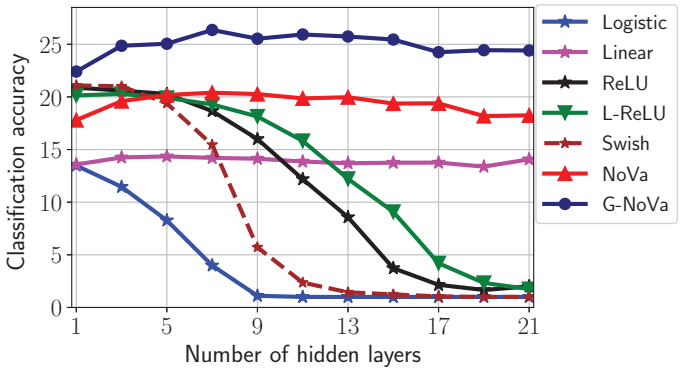


Fig. 6: Caltech-256 sample images: This figure shows 256 samples from the Caltech-256 dataset made up of 256 pattern classes.

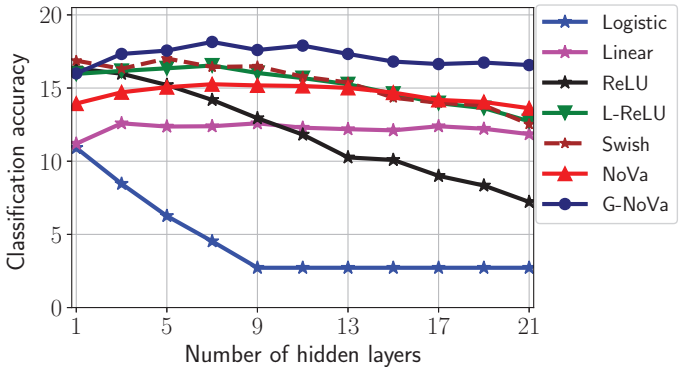
Unidirectional BP minimized the cross entropy at the output layer of softmax neurons. B-BP used the same activations and error function at the output layer. But it minimized the squared-error in the backward direction because the terminal input neurons have identity activations and so define a backward regressor. The neural classifiers trained over 100 epochs with stochastic gradient descent with learning rate $\alpha = 0.001$ and batch size $B = 64$.



(a) CIFAR-10 Dataset



(b) CIFAR-100 Dataset



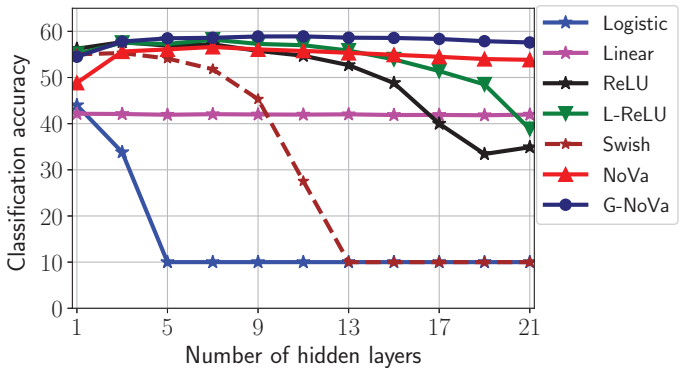
(c) Caltech-256 Dataset

Fig. 7: Benefit of hidden G-NoVa neurons in deep neural classifiers using unidirectional or ordinary BP: The models trained over 100 epochs with 500 neurons per hidden layer. The G-NoVa activation $a(x) = c\ x + x\sigma(b\ x)$ with $c = 0.5$ and $b = 1.0$ outperformed other activation functions where σ denotes logistic sigmoid. The comparative benefit of using hidden G-NoVa neurons increased as the depth of the neural classifiers increased.

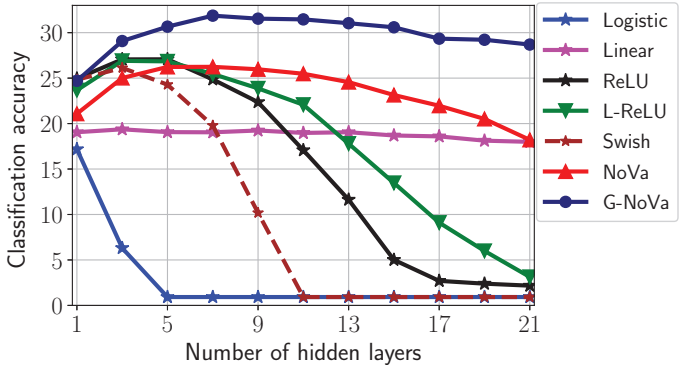
C. Results and Discussion

Figure 7 shows the classification-accuracy curves from simulations on the three datasets CIFAR-10, CIFAR-100, and Caltech-256. The neural classifiers trained with ordinary unidirectional BP. The models that used G-NoVa significantly outperformed leaky ReLU and other activations.

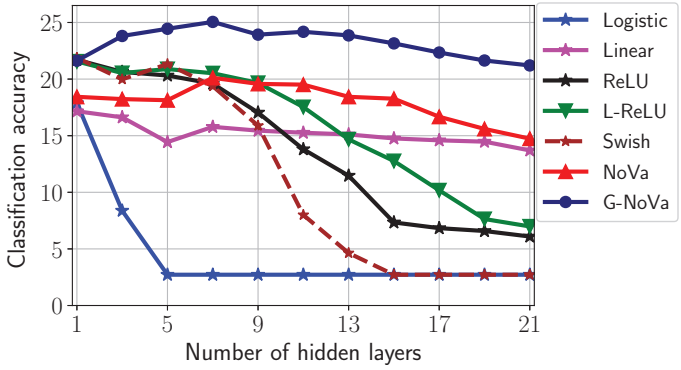
The benefit of G-NoVa grew as the depth of the network



(a) CIFAR-10 Dataset



(b) CIFAR-100 Dataset



(c) Caltech-256 Dataset

Fig. 8: Benefit of hidden G-NoVa neurons in deep neural classifiers using bidirectional BP: The models trained over 100 epochs using 500 neurons per hidden layer. G-NoVa activation $a(x) = c\ x + x\sigma(b\ x)$ with $c = 0.5$ and $b = 1.0$ outperformed other activation functions where σ denotes logistic sigmoid. The comparative benefit of using hidden G-NoVa neurons increased as the depth of the neural classifies increased.

TABLE II: Classification accuracy: Training deep neural classifiers with unidirectional BP algorithm on CIFAR-10 image dataset

Hidden Activation	3 Layers	11 Layers	21 Layers
Sigmoid	41.11%	10.00%	10.00%
Linear	35.24%	34.57%	34.87%
ReLU	50.36%	46.82%	26.74%
Leaky ReLU	49.61%	47.88%	29.22%
Swish	48.54%	17.04%	10.00%
NoVa	48.08%	48.16%	46.05%
G-NoVa	**51.80%**	**52.59%**	**51.55%**

TABLE III: Classification accuracy: Training deep neural classifiers with unidirectional BP algorithm on CIFAR-100 image dataset

Hidden Activation	3 Layers	11 Layers	21 Layers
Sigmoid	11.46%	1.00%	1.00%
Linear	14.25%	13.86%	14.07%
ReLU	20.60%	12.19%	2.00%
Leaky ReLU	20.29%	15.82%	1.75%
Swish	21.01%	2.36%	1.00%
NoVa	19.60%	19.86%	18.25%
G-NoVa	**24.86%**	**25.93%**	**24.41%**

TABLE IV: Classification accuracy: Training deep neural classifiers with unidirectional BP algorithm on Caltech-256 image dataset

Hidden Activation	3 Layers	11 Layers	21 Layers
Sigmoid	8.46%	2.72%	2.72%
Linear	12.59%	12.29%	11.85%
ReLU	15.98%	11.82%	7.22%
Leaky ReLU	16.17%	15.68%	12.73%
Swish	16.32%	15.78%	12.53%
NoVa	14.71%	15.14%	13.62%
G-NoVa	**17.33%**	**17.90%**	**16.75%**

TABLE V: Classification accuracy: Training deep neural classifiers with bidirectional BP algorithm on CIFAR-10 image dataset

Hidden Activation	3 Layers	11 Layers	21 Layers
Sigmoid	33.81%	10.00%	10.00%
Linear	42.09%	41.96%	41.96%
ReLU	57.67%	54.71%	34.89%
Leaky ReLU	57.52%	56.99%	38.83%
Swish	55.30%	27.54%	10.00%
NoVa	55.63%	58.91%	53.38%
G-NoVa	**57.79%**	**58.91%**	**57.59%**

increased. The consistent performance of G-NoVa is similar to what we noticed with the linear activation. The G-NoVa benefit tended to increase with big-K dataset images such as CIFAR-100 and Caltech-256 with respective $K = 100$ and $K = 256$. Tables II-IV show the same winning trend for G-NoVa networks.

Figure 8 shows the classification-accuracy curves from simulations on the three datasets using B-BP training algorithm. G-NoVa also outperformed other hidden activation functions. The models that used G-NoVa significantly outperformed leaky ReLU and other activations. G-NoVa benefit grows as the number of hidden layers increases. The G-NoVa benefit also tended to increase with big-K dataset images such as CIFAR-100 and Caltech-256 with respectivepattern numbers $K = 100$ and $K = 256$. Tables V-VII show the same trend.

TABLE VI: Classification accuracy: Training deep neural classifiers with bidirectional BP algorithm on CIFAR-100 image dataset

Hidden Activation	3 Layers	11 Layers	21 Layers
Sigmoid	6.31%	1.00%	1.00%
Linear	19.37%	18.97%	17.99%
ReLU	27.06%	17.07%	2.17%
Leaky ReLU	26.90%	22.07%	3.13%
Swish	26.13%	1.00%	1.00%
NoVa	25.01%	25.48%	18.21%
G-NoVa	**29.08%**	**31.46%**	**28.69%**

TABLE VII: Classification accuracy: Training deep neural classifiers with bidirectional BP algorithm on Caltech-256 image dataset

Hidden Activation	3 Layers	11 Layers	21 Layers
Sigmoid	8.38%	2.72%	2.72%
Linear	16.62%	15.26%	13.70%
ReLU	20.63%	13.80%	6.09%
Leaky ReLU	20.53%	17.53%	6.98%
Swish	19.98%	8.68%	2.72%
NoVa	18.23%	19.51%	14.74%
G-NoVa	**23.81%**	**24.17%**	**21.21%**

V. Conclusions

The new generalized nonvanishing (G-NoVa) hidden activation is a generalized perturbed logistic sigmoid whose derivative does not vanish. Simulations showed that it outperformed the simpler NoVa hidden neuron as well as the popular ReLU activation and its main variants. This comparative benefit in classification accuracy was most pronounced in the deepest networks. It held for both ordinary unidirectional and the newer bidirectional forms of backpropagation training. Future work will focus on testing G-NoVa hidden neurons on large datasets and convolutional models.

References

[1] O. Adigun and B. Kosko, "Deeper neural networks with non-vanishing logistic hidden units: Nova vs. relu neurons," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 1407–1412.

[2] J. Bauermann and B. Lindner, "Multiplicative noise is beneficial for the transmission of sensory signals in simple neuron models," *Biosystems*, vol. 178, pp. 25–31, 2019.

[3] A. Patel and B. Kosko, "Stochastic resonance in continuous and spiking neuron models with levy noise," *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 1993–2008, 2008.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[5] K. Fukushima, "Visual feature extraction by a multilayered network of analog threshold elements," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969.

[6] ——, "Cognitron: A self-organizing multilayered neural network," *Biological cybernetics*, vol. 20, no. 3, pp. 121–136, 1975.

[7] O. Adigun and B. Kosko, "Bidirectional backpropagation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 5, pp. 1982–1994, 2019.

[8] ——, "Bayesian bidirectional backpropagation learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–7.

[9] S. Grossberg, "Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world," *Neural networks*, vol. 37, pp. 1–47, 2013.

[10] L. E. B. da Silva, I. Elnabarawy, and D. C. Wunsch II, "A survey of adaptive resonance theory neural network models for engineering applications," *Neural Networks*, vol. 120, pp. 167–203, 2019.

[11] O. Adigun and B. Kosko, "Noise-boosted bidirectional backpropagation and adversarial learning," *Neural Networks*, vol. 120, pp. 9–31, 2019.

[12] B. Kosko, "Bidirectional associative memories: unsupervised hebbian learning to bidirectional backpropagation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 103–115, 2021.

[13] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural networks*, vol. 1, no. 1, pp. 17–61, 1988.

[14] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. USA: Prentice-Hall, Inc., 1992.

[15] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[16] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International workshop on artificial neural networks*. Springer, 1995, pp. 195–201.

[17] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[18] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Citeseer, 2013, p. 3.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[20] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[21] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[22] L. Lu, "Dying relu and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, 2020.

[23] L. Trottier, P. Giguere, and B. Chaib-Draa, "Parametric exponential linear unit for deep convolutional neural networks," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 207–214.

[24] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[25] O. Adigun and B. Kosko, "Bidirectional representation and backpropagation learning," in *International Joint Conference on Advances in Big Data Analytics*, 2016, pp. 3–9.

[26] ——, "Training generative adversarial networks with bidirectional backpropagation," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1178–1185.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[28] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[29] I. Goodfellow, Y. Bengio, and A. Courville, "Softmax units for multinoulli output distributions. deep learning," 2016.

[30] O. Adigun and B. Kosko, "Bidirectional backpropagation for high-capacity blocking networks," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 704–709.

[31] ——, "High capacity neural block classifiers with logistic neurons and random coding," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–9.

[32] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[33] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.