Received May 4, 2022, accepted May 24, 2022, date of publication May 30, 2022, date of current version June 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3178951

Two-Argument Activation Functions Learn Soft XOR Operations Like Cortical Neurons

JUHYEON KIM[1], EMIN ORHAN[2], KIJUNG YOON[1], AND XAQ PITKOW[3,4]
[1]Department of Electronic Engineering, Hanyang University, Seoul 04763, South Korea
[2]Center for Data Science, New York University, New York, NY 10011, USA
[3]Department of Neuroscience, Baylor College of Medicine, Houston, TX 77030, USA
[4]Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005, USA

Corresponding authors: Kijung Yoon (kiyoon@hanyang.ac.kr) and Xaq Pitkow (xaq@rice.edu)

This work was supported in part by the National Research Foundation of Korea (NRF) under Grant NRF-2018R1C1B5086404 and Grant NRF-2021R1F1A1045390; in part by the Brain Convergence Research Program of the National Research Foundation (NRF) through the Korean Government [Ministry of Science and ICT (MSIT)] under Grant NRF-2021M3E5D2A01023887; in part by the Technology Innovation Program (Development of Industrial Intelligent Technology for Manufacturing, Process, and Logistics) through the Ministry of Trade, Industry and Energy (MOTIE), South Korea, under Grant 20013726; in part by the Research Fund of Hanyang University under Grant HY-201900000000933; in part by NSF CAREER under Grant 1552868; in part by NSF NeuroNex under Grant 1707400; in part by the National Institutes of Health (NIH) BRAIN Initiative Grant under Grant 5U01NS094368; in part by the Award from the McNair Foundation; and in part by the Intelligence Advanced Research Projects Activity (IARPA) through the Department of Interior/Interior Business Center (DoI/IBC) under Contract D16PC00003.

ABSTRACT Neurons in the brain are complex machines with distinct functional compartments that interact nonlinearly. In contrast, neurons in artificial neural networks abstract away this complexity, typically down to a scalar activation function of a weighted sum of inputs. Here we emulate more biologically realistic neurons by learning canonical activation functions with two input arguments, analogous to basal and apical dendrites. We use a network-in-network architecture where each neuron is modeled as a multilayer perceptron with two inputs and a single output. This inner perceptron is shared by all units in the outer network. Remarkably, the resultant nonlinearities often produce soft XOR functions, consistent with recent experimental observations about interactions between inputs in human cortical neurons. When hyperparameters are optimized, networks with these nonlinearities learn faster and perform better than conventional ReLU nonlinearities with matched parameter counts, and they are more robust to natural and adversarial perturbations.

INDEX TERMS Biological and artificial neurons, activation functions, exclusive-or operation, adversarial robustness.

I. INTRODUCTION

Neurons in the brain are not simply linear filters followed by a half-wave rectification, and exhibit properties like divisive normalization [1], [2], coincidence detection [3], [4], and history dependence [5], [6]. Instead of fixed canonical nonlinear activation functions such as `sigmoid`, `tanh`, and `ReLU`, other nonlinearities may be both more realistic and more useful [7]–[9]. We are particularly interested in multivariate nonlinearities like $f(\boldsymbol{w}_1^\top \boldsymbol{x}, \boldsymbol{w}_2^\top \boldsymbol{x}, \ldots)$, where the arguments could correspond to inputs that arise, for example, from multiple distinct pathways such as feedforward, lateral, or feedback connections, or from different dendritic compartments. Such multi-argument nonlinearities could allow one feature to modulate the processing of the others.

Recent work showed that a single dendritic compartment of a single neuron can compute the exclusive-or (XOR) operation [10]. The fact that an artificial neuron could not compute this basic computational operation discredited neural networks for decades [11]. Although XOR can be computed by networks of neurons, the finding that even single neurons can also implement XOR highlights the possibility that individual neurons may be much more sophisticated than is often assumed in machine learning [9], [12]. Many single-argument nonlinearities permit universal computation, but the right nonlinearity could allow faster learning and better generalization, both for the brain and for artificial networks.

To investigate this, we parameterize the nonlinear input-output transformation flexibly by an "inner" neural network, which becomes a 'subroutine' called from the conventional "outer" network made of many of these

The associate editor coordinating the review of this manuscript and approving it for publication was Sunil Karamchandani.

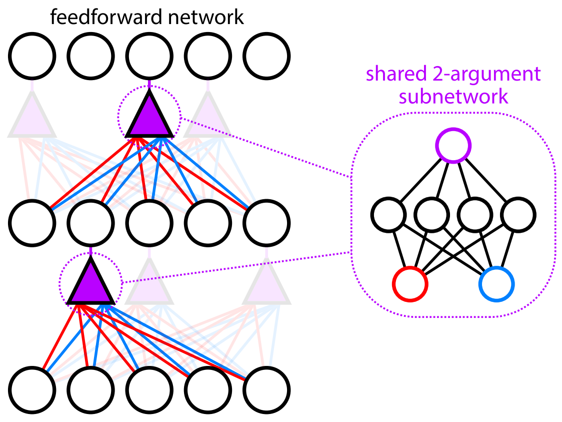


FIGURE 1. Multi-argument nonlinearities in artificial neurons. Schematic of architecture including a multi-argument nonlinear activation function (purple triangles). These functions' two arguments are different linear weighted sums of features, and may correspond to distinct inputs such as apical and basal dendrites.

complex neurons with parameters that are shared across all layers and all nodes of a given cell type (Figure 1). We evaluate fully-connected and convolutional feedforward networks on image classification tasks given a diverse set of random initial conditions. We focus especially on two-argument nonlinearities learned from MNIST and CIFAR-10 datasets.

II. RELATED WORK

Numerous recent studies have focused on developing novel activation functions, building on the simplicity and reliability of ReLU [13]–[16]. These studies can be distinguished by the type of learning algorithm used for optimizing the activation function and the size of the search space. Many recent modifications such as PReLU [17], ELU [18], SELU [19], GELU [20], RelTanh [21], and Isigmoid [22] provide single-argument activation functions with a small number of parameters. However, such hand-designed functional forms result in restricted expressivity. *Swish* [23] is noteworthy in this respect, because its activation function is discovered by a combination of exhaustive search and reinforcement learning. The search space in this case is based on a set of predetermined one- and two-argument functions, so this approach can span a broader class of nonlinearities than past work, although it is limited by the specific basis set and the combination rules chosen.

More closely related to our work, the network-in-network architecture proposes to replace groups of simple ReLUs with a fully connected network [24]. This activation function allows arbitrary dimensional inputs and outputs; thus it is essentially the most general and expressive nonlinear function. However, our work is primarily motivated by neurons in the brain, which can be formalized as multi-input and single-output nonlinear units. As in network-in-network, we parameterize the nonlinear many-to-one transformation by a fully-connected multi-layer network to examine the learned *spatial* activation function without sacrificing its representational power.

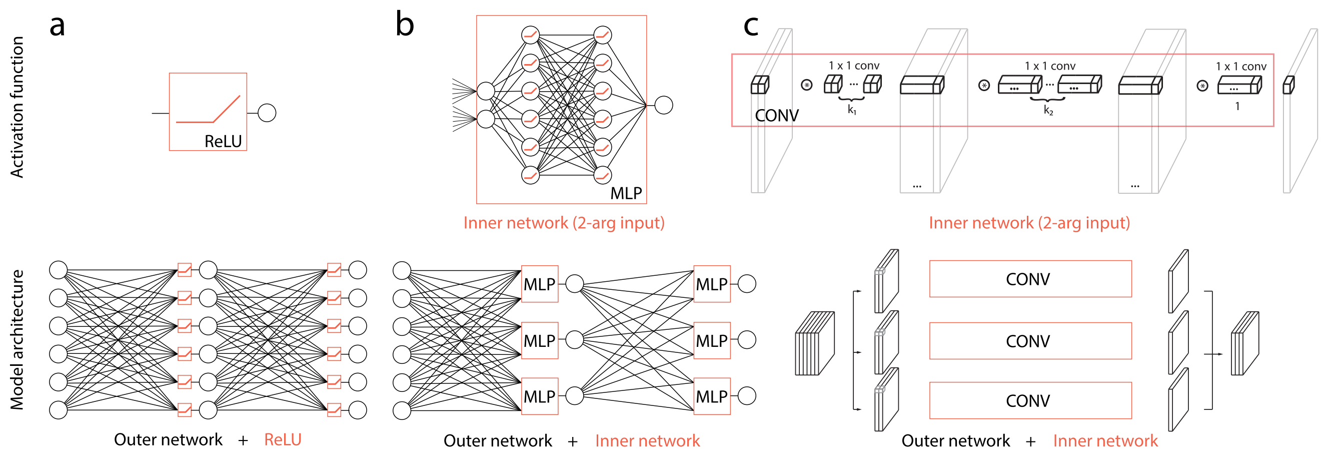
The multi-argument nonlinear transformation is also a canonical operation subsumed under the emerging network architectures such as graph neural networks (GNNs) [25]–[28] and transformers [29], [30]. As conceptual extensions from scalar to vector-valued inputs, the message functions in GNNs are multi-input nonlinearities while the scaled dot-product attention in transformers can be viewed as a three-input argument nonlinearity. Although these architectures evaluate performance benefits of specific multi-argument activations, to the best of our knowledge, ours is the first study to characterize the emergent properties of multivariate nonlinear activation functions and their connection to the neuronal nonlinearities in the brain.

III. MODEL STRUCTURE

To define our multi-argument nonlinearity, we introduce the concepts of inner network and outer network. The inner network aims to learn an arbitrary multivariate nonlinear function $f(x_1, \ldots, x_n)$ with n inputs and a single output. This will replace the regular scalar activation functions like ReLU. The outer network refers to the rest of the model architecture aside from the activation function. Our framework, composed of two disjoint networks, is flexible and general since diverse neural architectures can be used as outer networks, such as multilayer perceptrons (MLPs), convolutional neural networks (CNNs), ResNets, etc. On the other hand, for the inner network, we use MLPs that have two hidden layers with 64 units followed by ReLU nonlinearities. The MLP is shared across all layers, analogous to the fixed canonical nonlinear activation functions commonly used in feedforward deep neural networks. When we test a CNN-based outer network, we use 1×1 convolutions instead of MLPs for the inner network to make the model fully convolutional, but the inner network is otherwise the same as we used in the two-layer MLP. In this framework, the 1×1 conv implies that the inputs to the inner network are channel-wise features, which is similar to the idea of mixing channel information per location in the recent MLP mixer architecture [31]. Figure 2 summarizes how the inner network is incorporated into the outer network.

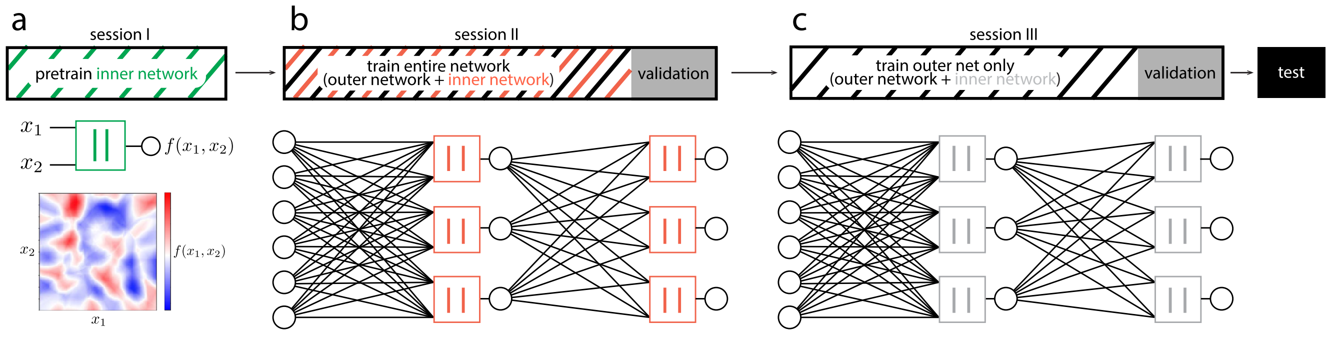
IV. EXPERIMENTS

To quantify the value of flexible multi-argument activation functions, we run a series of controlled experiments for different datasets and architectures. Our experiments have multiple training phases to ensure that we are isolating the consequences of the learned activation functions. We first pretrain a network to a random function to establish desired initial conditions, and then train both the inner and outer networks to determine the activation functions. With this activation function fixed, we then re-initialize and re-train the outer network, and only then do we evaluate its consequences. We compare performance to other activation functions, test



a
b
c
Activation function
Model architecture
ReLU
MLP
Inner network (2-arg input)
1 x 1 conv
CONV
Inner network (2-arg input)
Outer network + ReLU
Outer network + Inner network
CONV
Outer network + Inner network

FIGURE 2. **Overview of the proposed model structures. (a) Scalar nonlinear activation function ReLU (top) and MLP-based outer network with ReLU nonlinearities (bottom), (b) *n*-arg input MLP-based inner network (top; $n = 2$ in this figure) and the MLP-based outer network that replaces ReLU with the inner network above (bottom). The activation functions are color-coded by red boxes and the rest of the black other than the red boxes represents the elements of outer network, (c) 1 × 1 conv-based inner network (top) merged into conv-based outer network (bottom). The inner network takes inputs from different feature maps; thus the conv-based outer network requires slice and concatenation operations from the depth dimension before and after the inner network. The model schematics assume a two-input argument nonlinearity.**



a
session I
pretrain inner network
x_1
x_2
$f(x_1, x_2)$
b
session II
train entire network (outer network + inner network)
validation
c
session III
train outer net only (outer network + inner network)
validation
test

FIGURE 3. **Training procedure. (a) Pretraining. Schematic of two-input argument inner network (green) trained to predict a smoothed random initial activation map (bottom). (b) Simultaneously training inner (red) and outer (black) networks. (c) Retraining outer network (black) with frozen inner networks (gray).**

robustness to common corruptions and adversarial attack, and analyze the properties of the learned activation functions.

A. TRAINING PROCEDURE

1) PRETRAINING (SESSION I)

We first generate a random activation function and then use supervised learning to pretrain our inner network to match it (Figure 3a). The motivation for this inner network pretraining stage is to initialize the inner network to have a spatially complex nonlinearity as opposed to common initialization methods [32], [33], and to perform control experiments showing that even such a complex initialization quickly learns a smooth structure. To start with a sufficiently complex initial nonlinearity, we create a piecewise constant random output sampled uniformly from $[-1, 1]$ over a 5×5 grid of unit squares tiling the input space. We blur this by a 2D gaussian kernel ($\sigma = 3$ units) to define a random smooth activation map. This function serves as the target for the inner network to match (Figure 3a). Example activation functions after pretraining are shown in Figure 4b. This produces our initialized inner network, whose parameters are transferred to the next phase of training.

2) TRAINING INNER AND OUTER NETWORKS (SESSION II)

Next we merge the pretrained inner network with outer network via parameter sharing (Figure 3b) and apply this general network-in-network architecture to the task of image classification. In this session, both networks are trained simultaneously so that the entire network is made to learn over what might be analogous to an evolutionary timescale on which nonlinear cell properties emerge (Figure 3b). As our baseline outer networks, we use (1) MLPs that have three hidden layers with 64 units or (2) CNNs that have four convolutional layers with [60, 120, 120, 120] kernels of size 3×3 and a stride of 1, using 2×2 max-pooling with a stride of 2. Aside from the MLPs or convolutional layers, the outer network uses other standard architectural components: layer normalization [34] (placed before inner networks) and dropout [35] (placed after each hidden/convolutional layer; $p = 0.5$). Our models are trained on the MNIST and

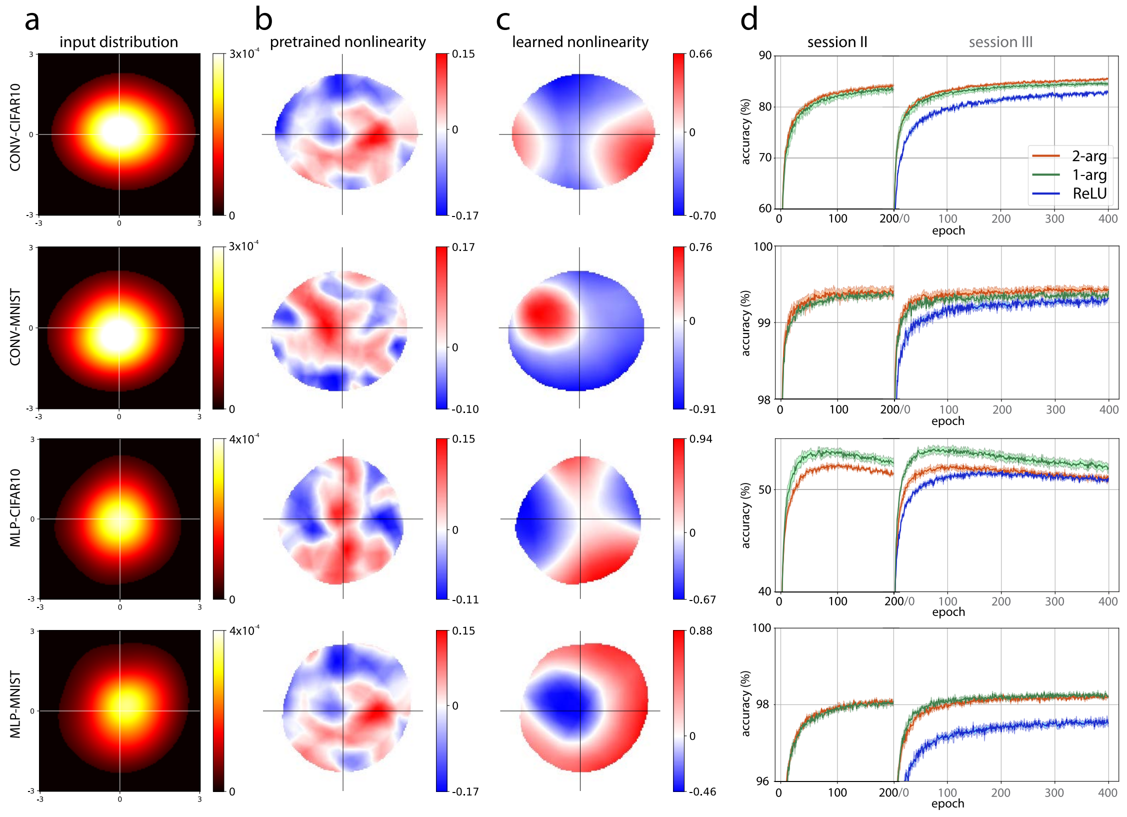


FIGURE 4. Learned nonlinearities learn tasks faster. Examples of (a) input distribution, (b) pretrained random initial nonlinearities, and (c) learned two-argument activation functions trained on two different data sets, CIFAR-10 and MNIST, within two different architecture types, a convolutional network and a multi-layer perceptron. Colors indicate the output of activation function, masked to the best-trained part of the input distribution, i.e. for the 99% of input values that are most common. White bands showing the crossing point between positive (blue) and negative (red) outputs. (d) Average test accuracy (solid line) ±1 SD (shaded region; $n = 4$ samples) of the 2-arg (red) and 1-arg (green) activation model and the baseline (blue: ReLU) in session II (200 epochs) and session III (400 epochs). Networks with these learnable nonlinearities learn faster than a fixed canonical nonlinear function.

CIFAR-10 datasets using ADAM [36] with a learning rate of 0.001 until the validation error saturates; early-stopping is used with a window size of 20. We freeze the learned nonlinearity $f_{\text{inner-net}}(\cdot)$ at the time of saturation or at a maximum epoch. Examples of learned nonlinearities are shown in Figure 4c.

To obtain some intuition about the learned 2-arg input nonlinearities, we first collect the values of every input to the nonlinearities (i.e. to the inner networks) over all test data at inference time. For display, we compute the pre-activation input distribution (Figure 4a), and show the nonlinearities over the region enclosing 99% of the input distribution (Figure 4b–c). If two-argument nonlinearities learned what is essentially a one-argument structure, we would see parallel bands of constant color. Instead, notably, all the examples show nontrivial two-dimensional structure, reflecting interactions between the two input arguments (see Section IV-C).

3) TRAINING OUTER NETWORK FOR FIXED INNER NETWORK (SESSION III)

Having learned multi-argument nonlinear activation functions, we now fix these inner networks and retrain the outer network to use them on new task data. We borrow the $f_{\text{inner-net}}(\cdot)$ from its parameters trained in session II, freeze the inner network, and then re-initialize the outer network. In this session, only the outer network is trained as for typical training of a deep neural network with a canonical activation function (Figure 3c). The training curves in this stage are not qualitatively different from what we observed in session II (Figure 4d), indicating that most of the learning over long time intervals (epochs) is attributable to the change of parameters in outer network. In other words, the learning of multi-argument nonlinear activation function may be terminated in an early stage and the rest of learning may be dedicated to solving the classification tasks.

We thus look for evidence of structural stability of inner network in early development by plotting the learned nonlinearities every epoch in session II. We find that the two-argument activation functions mature into typical two-dimensional spatial patterns within 1-5 epoch in general (Figure 5), suggesting that the overall spatial structure of the activation function emerges quite rapidly from pressures that arise early in the learning process.

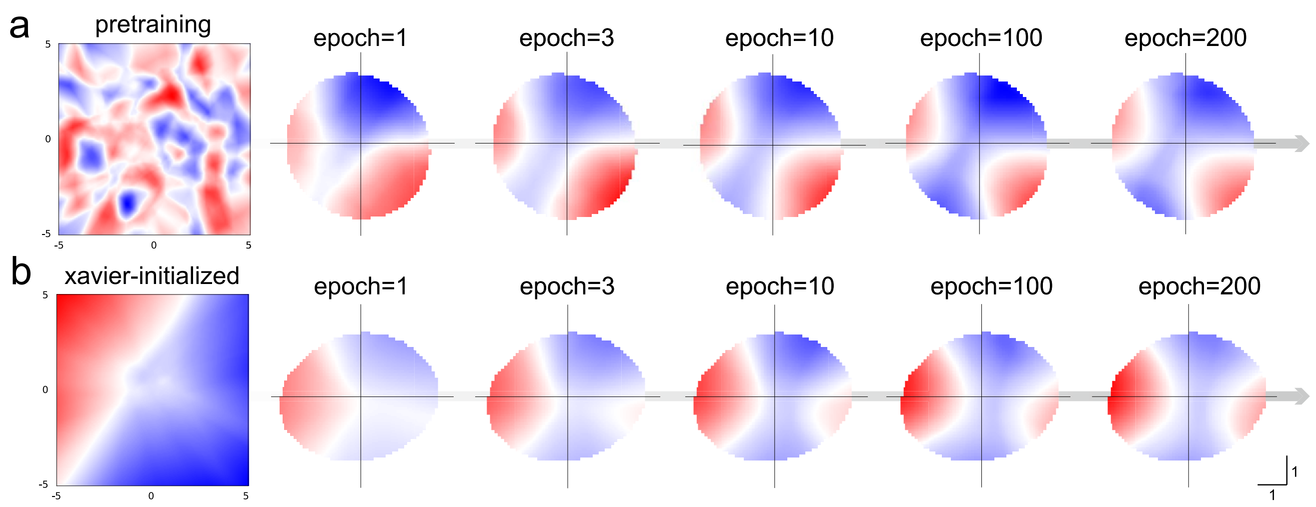


FIGURE 5. **Evolution of learned two-argument activation functions. (a) Snapshot of random initial and learned nonlinear activation functions across development. (b) The same evolution of nonlinearity when it is Xavier-initialized.**

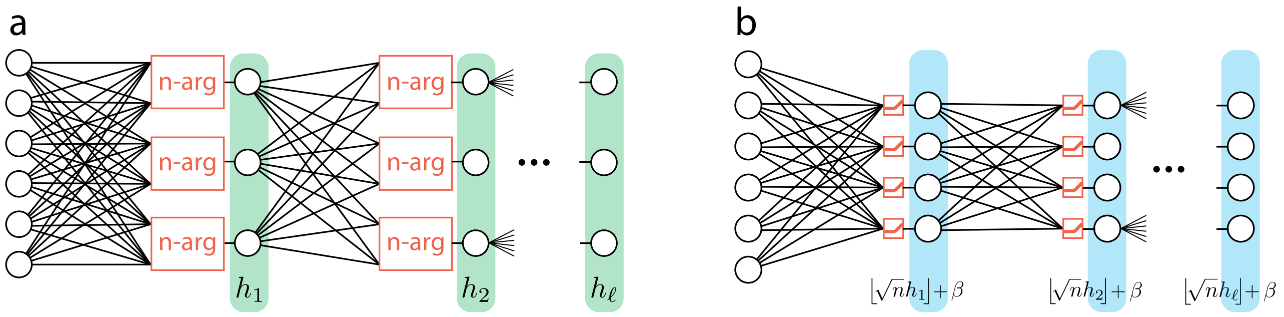

a
n-arg
h_1
h_2
h_ℓ
b
$\lfloor\sqrt{n}h_1\rfloor+\beta$
$\lfloor\sqrt{n}h_2\rfloor+\beta$
$\lfloor\sqrt{n}h_\ell\rfloor+\beta$

FIGURE 6. **Baseline architecture for parameter counts. (a) MLP-based outer network that have L hidden layers with h_ℓ units (green) along with n-arg input nonlinearities (red). (b) Baseline model architecture with ReLU composed of L hidden layers with $\lfloor\sqrt{n}h_\ell\rfloor + \beta$ units (blue) in each layer ℓ.**

B. COMPARING TO OTHER NONLINEARITIES

With the aim of providing context for the performance of our proposed approach we compare against a single-argument nonlinearity. For fair comparison, we train the baseline models, whose architectures are depicted in Figure 6, just as we train our outer networks. The baseline models all involve the same MLP or CNN architecture, i.e. they use the same type and number of outer network layers as our proposed model. We keep the depths of our networks matched, even though the inner network nominally creates additional layers, because that inner network is frozen during our second phase of training. This approach allows us to compare the consequences of our new activation functions to other more conventional ones.

When comparing different architectures we take care to use comparable numbers of learnable parameters in the classification tasks by systematically adjusting the number of hidden units or feature maps in each layer. Specifically, the MLP-based outer network with n-arg input nonlinearities (Figure 5a) contains $x(nh_1 + 1) + \sum_{\ell=1}^{L-1} nh_\ell h_{\ell+1} + h_L y + (65n+4288)$ parameters, where x, y, and h_ℓ are the number of inputs, outputs, and units in hidden layer ℓ, respectively. The last term represents the number of inner network parameters; this is independent of input and output dimensions as well as the number of hidden layers L, so it does not increase the model complexity (due to parameter sharing). In contrast, the second term $\sum_{\ell=1}^{L-1} nh_\ell h_{\ell+1}$ dominates the parameter count. We fix the number of layers to L for all networks, but scale the width of each layer to maintain the same parameter count for the outer networks (Figure 6b). We scale all layer widths by the same global factor so the relative profile of layer widths $h_\ell / \sum_k h_k$ are preserved for every network. This scaling factor is $\sqrt{n}$ for an n-arg nonlinearity, because the weights scale as the product of layer widths: $\lfloor\sqrt{n}h_\ell\rfloor \times \lfloor\sqrt{n}h_{\ell+1}\rfloor \approx nh_\ell h_{\ell+1}$. To compensate for the different scaling from the fixed input layer, and for the small discrepancy between this desired irrational scaling and the integer numbers of hidden units, we adjust the widths by a small additive term β. To match parameter counts across networks, each layer thus comprises $\lfloor\sqrt{n}h_\ell\rfloor + \beta$ hidden units. This way of matching parameter counts in MLP-based outer network applies also to CNN-based models, by setting h_ℓ to be the number of feature maps in convolutional layer ℓ instead of hidden units.

Figure 4d compares training performance of the two-input argument nonlinearity to networks using a ReLU or single-argument nonlinearity. We repeat the training of the

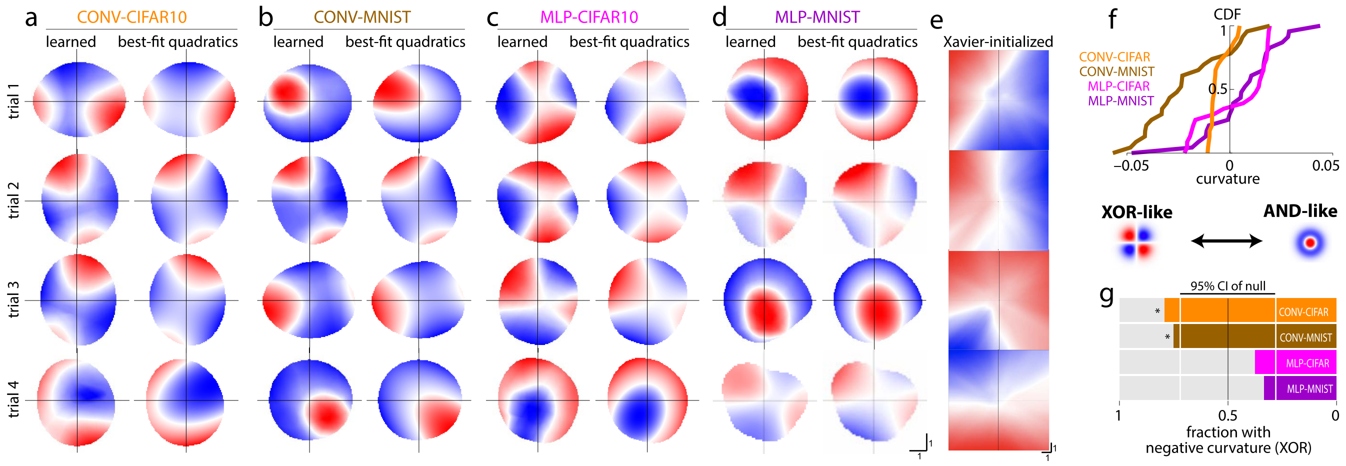


FIGURE 7. **Gating operations emerge naturally from learnable multi-argument nonlinear structures. (a-d) Left: Examples of learned multi-argument activation functions trained on CIFAR-10 and MNIST, within two different architecture types, CNN and MLP. Each row is a different repetition of the learning experiment. All examples show nontrivial two-dimensional structure, reflecting interactions between two input arguments. The majority show a (potentially rotated) white X shape, indicating a multiplicative interaction between the input features, and consistent with a gating interaction or soft XOR. (a-d) Right: The best-fit quadratics of the corresponding left nonlinearities. (e) Random activation functions generated from Xavier weight initialization. (f) Cumulative Distribution Function (CDF) of nonlinearity curvature. (g) Fraction of nonlinearities with negative (XOR-like) curvature. Even a set of random functions may by chance have nonzero average curvature. The CONV architectures show deviations that are outside of the 95% Confidence Interval (CI) of the null distribution (binomial distribution with probability of 1/2 for positive or negative curvature, for 24 trials).**

nonlinearities on MNIST and CIFAR-10 4 times, which produces 4 different samples of model performance. We average the results across 4 samples and find that the models with learned activation functions achieve an overall strong performance (Figure 4d). Notably, at any given epoch in Figure 4d, performance of the learned nonlinearities is better than the performance of the baseline ReLU nonlinearity. The result suggests that our proposed network learns faster than the ReLU network and achieves better asymptotic performance, providing evidence for a better inductive bias in the network due to the learned multi-argument nonlinearities.

C. EXPLICIT POLYNOMIAL NONLINEARITIES

The results outlined in the previous section focus on the predictive performance of multivariate nonlinear functions. We next turn our attention to the analysis of the structure learned by our multi-argument nonlinearities. We repeat four different trials of the learning experiment and collect samples of two-argument activation functions trained on MNIST and CIFAR-10, within MLP and CNN outer networks. Figure 7a–d (left columns) demonstrates that learned two-argument nonlinearities are reliably shaped like quadratic functions, varying by shifts and/or rotations. We therefore fit an algebraic quadratic functional form, $f(x_1, x_2) = c_1x_1^2 + c_2x_2^2 + c_3x_1x_2 + c_4x_1 + c_5x_2 + c_6$, to the learned inner-network nonlinearities and find that the learned nonlinearity and its best-fit quadratics have extremely similar structure (Figures 7a–d right). This is the case even though the spatial patterns have different rotations (Figures 7a–d).

We next validate the specificity of the observed inner network output responses. It is clear by eye that the learned nonlinearities are substantially different than those produced by random functions (Figure 4b–c). However, this regular pattern of learned nonlinearities might also be obtainable by popular network initialization methods, such as Xavier weight initialization. To differentiate between these two possibilities, we therefore compare the learned nonlinearities with inner nets initialized with Xavier random initialization [32] (Figure 7e). We find that the Xaiver random initial activations, although not as "random" as those we generated ourselves (Figure 4b), are far from the regular quadratic patterns observed in the learned nonlinearities (Figure 7e). They instead evolve to display such smooth quadratic patterns (Figure 5b), suggesting that the quadratic structures we observe are not captured by standard weight initialization schemes, but are favored by the optimization process instead.

To test whether the learned quadratic functions have statistically significant sub-structure (for example, hyperbolic vs. elliptical or negative vs. positive curvature), we computed the curvature implied by the quadratic form above, $c_1c_2 - c_3^2/4$ (Figure 7f–g). The convolutional architecture learned nonlinearities with negative curvatures for both tasks, a total of 78% of 48 trials ($p = 0.007$ according to a binomial null distribution with even odds of either curvature). This indicates a multiplicative interaction between the input features, and is consistent with a gating interaction or soft XOR. In contrast, the multilayer perceptron architecture produced more positive curvatures, but these were not statistically significant ($p = 0.06$ by the same test).

D. SPECTRAL ANALYSIS

In principle, these multi-argument nonlinearities could have learned a one-argument function, with a response that is invariant to a second filter dimension. This would essentially replicate the structure of a pure one-argument activation

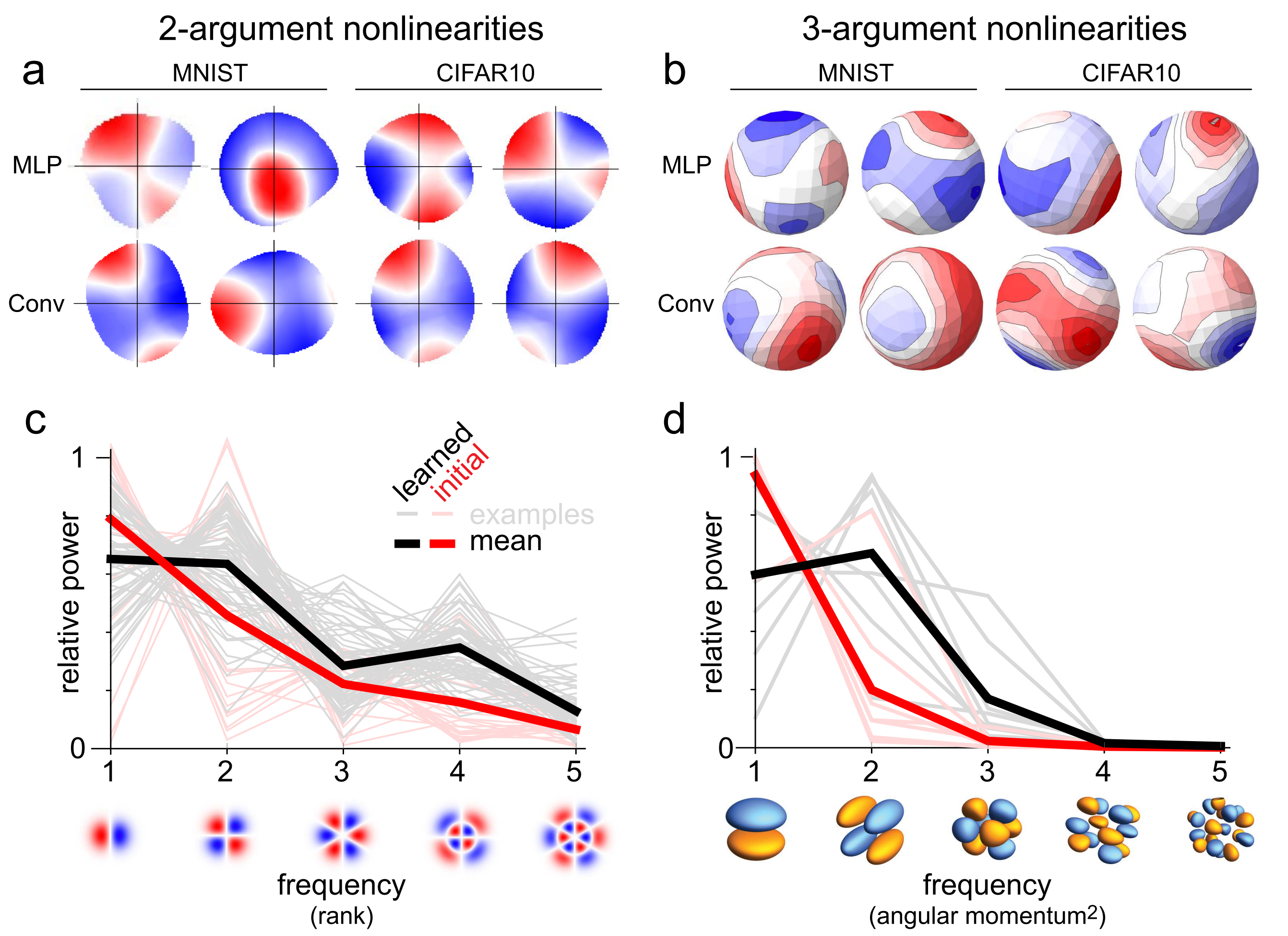


FIGURE 8. Spectral Analysis. Nonlinearities for various architectures and tasks for (a) two-argument and (b) three-argument inner networks. (c–d) Power spectra for these learned functions (black curves) reveal larger power at $\ell = 2$ than spectra for Xavier-initialized inner networks (red), consistent with stronger quadrupolar structure. For the two-argument case, we used 64 learned functions and 24 randomly initialized functions. For the three-argument case, we used 8 learned functions for each. Example basis functions are shown beneath the horizontal axis to illustrate the spatial structure quantified by the frequency number.

function. However, this would waste many of the degrees of freedom that the outer network could use for learning useful projections of its input layer. We selected our architectural hyperparameters to match the total number of parameters for one-argument and multi-argument cases, so if a unit's response is invariant to some of its input projections, then the one-argument nonlinearity would have been able to capture that structure just as well, but with more distinct filters. The multidimensional nonlinearity therefore is incentivized to produce nontrivial interactions between its input arguments.

To characterize the structure of learned nonlinearities, we performed a spectral analysis on them. We then compared the spectra of learned nonlinearities to those of Xavier-initialized ones. Additionally, to check the generality of these results, we also optimized *three*-argument activation functions and compared them with the two-argument case. This allowed us to examine whether the quadratic structure is preserved even in higher-order input argument nonlinearities.

We computed spectra using basis functions $\phi(\boldsymbol{x})$ appropriate for the symmetry and boundary conditions of the nonlinearities: we used Hermite-Bessel functions [37] for the two-argument functions, and solid harmonics for three-argument activation functions. We only evaluated the power in regions of the input space that were explored by the distribution $p(\boldsymbol{x})$ of their actual inputs. The power was therefore computed according to $P_\ell[f(\boldsymbol{x})] = \sum_m \| \int d\boldsymbol{x}\, p(\boldsymbol{x}) f(\boldsymbol{x}) \phi_{\ell m}(\boldsymbol{x}) \|^2$, where ℓ is the analog of spatial frequency for these basis functions and m is analogous to spatial phases.

Figure 8 shows that the learned multi-argument nonlinearities have more higher-order structure than the Xavier initialized ones. Randomly initialized networks favor strong dipole structure with $\ell = 1$. In contrast, the power spectra of learned nonlinearities are consistent with an underlying quadrupole structure, which has its strongest frequency content at $\ell = 2$. A soft XOR can be described by $f(x_1, x_2) = x_1 x_2$ or its rotations, which produces positive outputs in two opposite quadrants and therefore creates a quadrupole moment with negative curvature.

E. GENERALIZATION

We now consider out-of-distribution generalization performance of the models for image classification with multi-argument nonlinear functions. In particular, we test whether these activation functions make the learned representations more robust against common image corruptions and adversarial perturbations. We quantify the robustness of the models against common corruptions and perturbations using the recently introduced CIFAR-10-C benchmark [38] and AutoAttack [39], a parameter-free ensemble of black- and white-box attacks. We refer to [38] and [39] for their description of how these benchmark datasets and attacks were developed.

1) ROBUSTNESS AGAINST COMMON IMAGE CORRUPTIONS

CIFAR-10-C was designed to measure the robustness of classifiers against common image corruptions and contains 15 different corruption types applied to each CIFAR-10 validation image at 5 different severity levels. The robustness performance on CIFAR-10-C is measured by the corruption error (CE). For each corruption type c, the classification error of the two-argument network is averaged over different severity levels s and then divided by the average classification error of a reference classifier (conv-based outer network with ReLU): i.e. $\text{CE}_c^{\text{2-arg}} = \left(\sum_{s=1}^{5} E_{s,c}^{\text{2-arg}}\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\text{ReLU}}\right)$. The mean corruption error is then obtained by averaging over the corruption types: $\text{mCE} = \langle \text{CE}_c^{\text{2-arg}} \rangle_c$. We also compute a relative mCE score by subtracting the clean classification error of the classifiers from the corruption errors: Relative $\text{CE}_c^{\text{2-arg}} = \left(\sum_{s=1}^{5} E_{s,c}^{\text{2-arg}} - E_{\text{clean}}^{\text{2-arg}}\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\text{ReLU}} - E_{\text{clean}}^{\text{ReLU}}\right)$ and then averaging over different corruption types as before results in the relative mCE $= \langle \text{relative CE}_c^{\text{2-arg}} \rangle_c$. This measures the relative enhancement on corrupted images in comparison with clean images.

As seen in Figure 9, two-input argument nonlinearities significantly improve the robustness over the ReLU baseline model (mCE $= 91.3\%$). Note that mCE scores lower than 100 indicate more success at generalizing to corrupted distribution than the reference model. Moreover, the observed relative mCE ($= 99.5\%$, which is less than 100) shows that the accuracy decline of the proposed model in the presence of corruptions is on average less than that of the network with ReLU. The results suggest that this corruption robustness improvements be attributable not only to the simple model accuracy improvements on clean images, but to stronger representations of the learnable multivariate nonlinearity than ReLU against natural corruptions. These relative measure are standard performance metrics chosen to highlight real robustness benefits, properly accounting for confounds like overall performance [38].

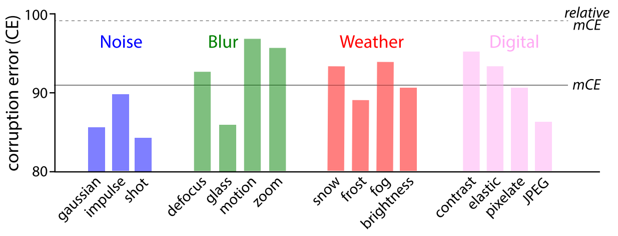


FIGURE 9. Robustness of two-argument nonlinearities against common image corruptions. Corruption error (CE; bars), mCE (black solid line), and relative mCE (black dashed line) of different corruptions on CIFAR-10-C and Conv-based outer networks. The mCE is the mean corruption error of the corruptions in Noise, Blur, Weather, and Digital categories. Models are trained only on clean CIFAR-10 images.

TABLE 1. Robustness of adversarial defenses by *AutoAttack*. Numbers indicate average classification accuracy from 4 trials.

		AutoAttack			
Dataset	Outer-Net	2-arg	1-arg	ReLU	increment
MNIST ($l_\infty, \epsilon = 0.3$)	MLP	39.80	22.86	26.74	13.06
MNIST ($l_\infty, \epsilon = 0.3$)	Conv	49.25	10.02	9.33	39.92
CIFAR-10 ($l_\infty, \epsilon = 0.031$)	MLP	4.83	5.62	2.96	1.87
CIFAR-10 ($l_\infty, \epsilon = 0.031$)	Conv	11.27	9.55	8.57	2.70

2) ADVERSARIAL ROBUSTNESS

We next consider both black-box and white-box attacks to measure the robustness of the model against adversarial perturbations. We use the recently introduced AutoAttack [39] combining two parameter-free versions of Projected Gradient Descent (PGD) [40] algorithm with two existing complementary Fast Adaptive Boundary (FAB) [41] and Square [42] attacks. AutoAttack is carried out with an ensemble of the four aforementioned attacks to reliably evaluate adversarial robustness where the hyperparameters of all attacks are fixed for all experiments across datasets and models.

In Table 1, we report the results on 6 models ($\in \{\text{MLP, Conv}\}_{\text{outer-net}} \times \{\text{2-arg, 1-arg, ReLU}\}_{\text{inner-net}}$) trained for ℓ_∞-robustness. For each classifier we report the accuracy on the robustness test, at the ϵ specified in the table, on the whole test set obtained by the ensemble AutoAttack. This method counts an attack successful when at least one of the four attacks finds an adversarial example (worst case evaluation). Additionally, we compute the difference in robustness between the network with two-input argument nonlinearities and the baseline model using ReLU nonlinearities. Positive differences are highlighted in blue in the last column of Table 1, and indicate improved robustness compared to the baseline model. In all cases, AutoAttack reveals greater robustness in networks with the learned two-argument nonlinearities than in the baseline networks with ReLU. This suggests that the learned two-argument nonlinearities provide a better inductive bias against adversarial perturbations.

V. DISCUSSION

The neurons in biological neural networks are much more intricate machines than the units they inspired in machine learning. Instead, neural networks in machine learning have been dominated by scalar activation functions. At the same time, it is widely acknowledged that different design choices here can lead to different inductive biases, and architectures with new neural elements are proposed frequently. These elements are usually based on guesses or intuition. Interestingly, one of the most influential elements has been a multiplicative gating nonlinearity, seen in LSTMs [43], GRUs [44], and transformers [29]. Our experiments demonstrated that gating-like functions emerge automatically from learned multi-argument nonlinear activation functions, as the soft XOR can be interpreted as an output that selects one input dimension of its input and modulates or gates that output by another input dimension. These learned functions have properties resembling dendritic interactions in biological neurons [10]. Networks endowed with these functions learn faster and are more robust.

Although these learnable nonlinearities add some complexity to a network, overall these extra inner network parameters are few in number since they are shared across all neurons in the outer network. Moreover, using algebraic polynomial approximations to the learned nonlinearities, as in section IV-C, can reduce both the number of parameters and the memory requirements of the inner networks in practical applications.

Nontrivial computations in a multilayer network require some sort of nonlinearity, since otherwise the whole network merely performs one linear transformation. The simplest nonlinearity is quadratic, whether the quadratic has negative curvature like a soft XOR, or a positive curvature like coincidence detection. It is interesting that even when allowing for more input arguments, the resultant learned nonlinearities still favor low-order quadratic functions (Figure 8b–d). This could be explained by an implicit bias toward smooth functions [45], [46] while still bending the input space to provide useful computations. Perhaps the learned nonlinearities are as random as possible while fulfilling these minimal conditions. It will be interesting to test this hypothesis by examining the transformations of multiple cell types, or those produced by higher-dimensional functions like network-in-network [24], and to see whether different tasks incentivize different computations.

VI. CONCLUSION

Our study demonstrates that flexible multi-argument activation functions converge to reliable and interpretable patterns and provide computational benefits. However, our study has important limitations that should be addressed in future work. The performance benefits should be evaluated in more architectures and tasks, and at larger scales. There might be synergistic benefits from additional features like skip connections or global modulation. Some of the additional complexity afforded by multi-argument activation functions might be more useful when used in richer architectures, including those with recurrence, dedicated input types (e.g. distinct feedforward, feedback, and lateral interaction arguments), multiple cell types [47], [48], and more intricate dendritic substructures [7], [12]. Such biologically-inspired additions to neural network architectures could provide inductive biases closer to the inductive biases in biological brains [49], [50].

ACKNOWLEDGMENT

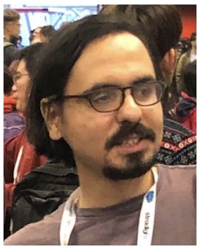
The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: the views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

REFERENCES

[1] D. J. Heeger, "Normalization of cell responses in cat striate cortex," *J. Neurosci.*, vol. 9, no. 2, pp. 181–197, 1992.
[2] M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nature Rev. Neurosci.*, vol. 13, pp. 51–62, Jan. 2012.
[3] M. E. Larkum, J. J. Zhu, and B. Sakmann, "A new cellular mechanism for coupling inputs arriving at different cortical layers," *Nature*, vol. 398, no. 6725, pp. 338–341, Mar. 1999.
[4] T. Branco, B. A. Clark, and M. Häusser, "Dendritic discrimination of temporal input sequences in cortical neurons," *Science*, vol. 329, no. 5999, pp. 1671–1675, Sep. 2000.
[5] H. B. Barlow, "Possible principles underlying the transformation of sensory messages," *Sensory Commun.*, vol. 1, no. 1, pp. 217–234, 1961.
[6] F. Rieke and D. Warland, *Spikes: Exploring the Neural Code*. Cambridge, MA, USA: MIT Press, 1999.
[7] P. Poirazi, T. Brannon, and B. W. Mel, "Pyramidal neuron as two-layer neural network," *Neuron*, vol. 37, no. 6, pp. 989–999, Mar. 2003.
[8] D. Beniaguev, I. Segev, and M. London, "Single cortical neurons as deep artificial neural networks," *Neuron*, vol. 109, no. 17, pp. 2727–2739, 2021.
[9] I. S. Jones and K. P. Kording, "Might a single neuron solve interesting machine learning problems through successive computations on its dendritic tree?" *Neural Comput.*, vol. 33, no. 6, pp. 1554–1571, May 2021.
[10] A. Gidon, T. A. Zolnik, P. Fidzinski, F. Bolduan, A. Papoutsi, P. Poirazi, M. Holtkamp, I. Vida, and M. E. Larkum, "Dendritic action potentials and computation in human layer 2/3 cortical neurons," *Science*, vol. 367, no. 6473, pp. 83–87, Jan. 2020.
[11] M. Minsky and S. Papert, *Perceptrons*. Cambridge, MA, USA: MIT Press, 1969.
[12] P. Poirazi and B. W. Mel, "Impact of active dendrites and structural plasticity on the memory capacity of neural tissue," *Neuron*, vol. 29, no. 3, pp. 779–796, Mar. 2001.
[13] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, Jun. 2000.
[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.
[15] B. Hanin, "Universal function approximation by deep neural nets with bounded width and ReLU activations," *Mathematics*, vol. 7, p. 992, Oct. 2019.
[16] J. Whittington and R. Bogacz, "Theories of error back-propagation in the brain," *Trends Cogn. Sci.*, vol. 23, no. 3, pp. 235–250, 2019.
[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[18] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.
[19] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 972–981.
[20] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[21] Y. Qin, X. Wang, and J. Zou, "The optimized deep belief networks with improved logistic Sigmoid units and their application in fault diagnosis for planetary gearboxes of wind turbines," *IEEE Trans. Ind. Electron.*, vol. 66, no. 5, pp. 3814–3824, May 2018.
[22] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen, "ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis," *Neurocomputing*, vol. 363, pp. 88–98, Oct. 2019.
[23] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.
[24] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.
[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.
[26] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*.
[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[28] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017.
[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[30] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," 2021, *arXiv:2103.03206*.
[31] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "MLP-mixer: An all-MLP architecture for vision," 2021, *arXiv:2105.01601*.
[32] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[34] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdi, "DropOut: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Nov. 2014.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[37] J. D. Victor, F. Mechler, M. A. Repucci, K. P. Purpura, and T. Sharpee, "Responses of v1 neurons to two-dimensional Hermite functions," *J. Neurophysiol.*, vol. 95, no. 1, pp. 379–400, Jan. 2006.
[38] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.
[39] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.
[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2017.
[41] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2196–2205.
[42] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 484–501.
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.
[45] F. Williams, M. Trager, C. Silva, D. Panozzo, D. Zorin, and J. Bruna, "Gradient dynamics of shallow univariate ReLU networks," 2019, *arXiv:1906.07842*.
[46] J. Sahs, R. Pyle, A. Damaraju, J. O. Caro, O. Tavaslioglu, A. Lu, and A. Patel, "Shallow univariate ReLu networks as splines: Initialization, loss surface, hessian, & gradient flow dynamics," 2020, *arXiv:2008.01772*.
[47] R. J. Douglas and K. A. Martin, "A functional microcircuit for cat visual cortex," *J. Physiol.*, vol. 440, no. 1, pp. 735–769, Aug. 1991.
[48] G. M. Shepherd, *The Synaptic Organization of the Brain*. Oxford, U.K.: Oxford Univ. Press, 2004.
[49] F. H. Sinz, X. Pitkow, J. Reimer, M. Bethge, and A. S. Tolias, "Engineering a less artificial intelligence," *Neuron*, vol. 103, no. 6, pp. 967–979, 2019.
[50] A. Litwin-Kumar and S. C. Turaga, "Constraining computational models using electron microscopy wiring diagrams," *Current Opinion Neurobiol.*, vol. 58, pp. 94–100, Oct. 2019.

JUHYEON KIM received the B.S. degree in electronic and electrical engineering from Hongik University, Seoul, South Korea, in 2021. He is currently pursuing the M.S. degree with the Department of Electronic Engineering, Hanyang University, Seoul. His research interests include machine learning, deep learning, artificial intelligence, and their applications.



EMIN ORHAN received the M.S. degree in cognitive science from Boğaziçi University, Istanbul, Turkey, in 2008, and the Ph.D. degree in brain and cognitive sciences from the University of Rochester, Rochester, NY, USA, in 2013. From 2013 to 2016, he was a Postdoctoral Researcher at the Center for Neural Science, New York University, NY, USA. From 2016 to 2019, he was a Postdoctoral Researcher at the Department of Neuroscience, Baylor College of Medicine. He is currently a Postdoctoral Researcher with the Center for Data Science, New York University. His current research interests include machine learning, artificial intelligence, computer vision, natural language processing, cognitive science, and computational neuroscience.

KIJUNG YOON received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2008, and the M.S. and Ph.D. degrees in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 2011 and 2015, respectively. From 2016 to 2018, he was a Postdoctoral Researcher at the Department of Neuroscience, Baylor College of Medicine, and the Department of Electrical and Computer Engineering, Rice University. From 2018 to 2019, he was an Assistant Professor at the School of Electronic and Electrical Engineering, Hongik University, Seoul, South Korea. He is currently an Assistant Professor with the Department of Electronic Engineering, Hanyang University, Seoul. His current research interests include machine learning, artificial intelligence, and computational neuroscience.

XAQ PITKOW received the B.S. degree in physics from Princeton University, Princeton, NJ, USA, in 1997, and the Ph.D. degree in biophysics from Harvard University, Cambridge, MA, USA, in 2006. From 2007 to 2010, he was a Postdoctoral Research Fellow at the Center for Theoretical Neuroscience, Columbia University. From 2010 to 2013, he was a Postdoctoral Research Scientist at the University of Rochester. He is currently an Associate Professor with the Department of Neuroscience, Baylor College of Medicine, and the Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA. His current research interests include theoretical and computational neuroscience, machine learning, and artificial intelligence.