Two-argument activation functions learn soft XOR operations like cortical neurons

Kijung Yoon*
Department of Electronic Engineering
Hanyang University

Emin Orhan
Center for Data Science
NYU

Juhyun Kim
Department of Electronic Engineering
Hanyang University

Xaq Pitkow*
Department of Neuroscience
Baylor College of Medicine
Department of ECE
Rice University

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.

1 Introduction

Neurons in the brain are not simply linear filters followed by a half-wave rectification, and exhibit properties like divisive normalization (Heeger, 1992; Carandini and Heeger, 2012), coincidence detection (Larkum et al., 1999; Branco et al., 2010), and history dependence (Barlow et al., 1961; Rieke and Warland, 1999). Instead of fixed canonical nonlinear activation functions such as `sigmoid`, `tanh`, and `ReLU`, other nonlinearities may be both more realistic and more useful (Poirazi et al., 2003; Beniaguev et al., 2021; Jones and Kording, 2021). We are particularly interested in multivariate nonlinearities like $f(\boldsymbol{w}_1^\top \boldsymbol{x}, \boldsymbol{w}_2^\top \boldsymbol{x}, ...)$, 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.

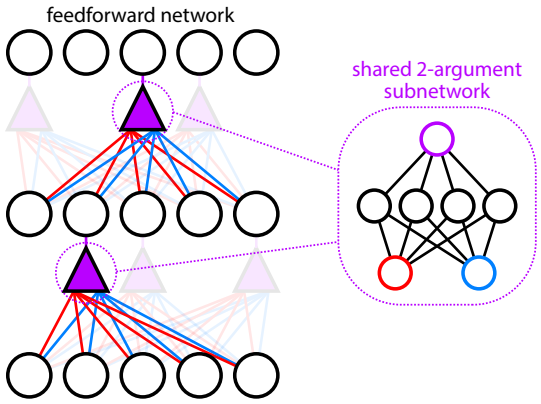

Figure 1: **Multi-argument nonlinearities in artificial neurons.** Schematic of architecture including a multi-argument nonlinear activation function (purple triangles). These functions' two arguments are different linear weighted sums of features, and may correspond to distinct inputs such as apical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

Recent work showed that a single dendritic compartment of a single neuron can compute the exclusive-or (XOR) operation (Gidon et al., 2020). The fact that an artificial neuron could not compute this basic computational operation discredited neural networks for decades (Minsky and Papert, 1969). Although XOR

*Correspondence to:
<kiyoon@hanyang.ac.kr>, <xaq@rice.edu>

Preprint. Under review.

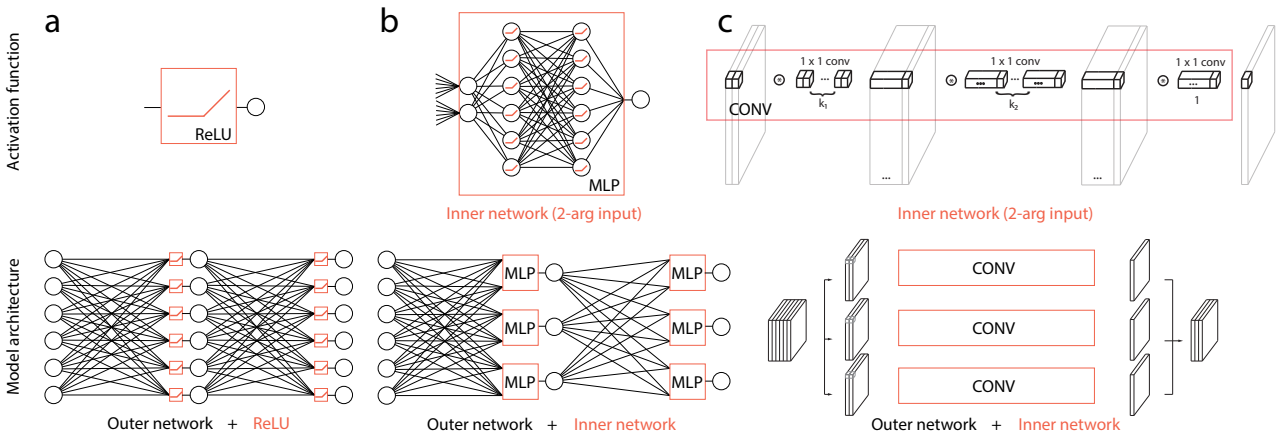

Figure 2: **Overview of the proposed model structures. (a)** Scalar nonlinear activation function ReLU (top) and MLP-based outer network with ReLU nonlinearities (bottom), **(b)** n-arg input MLP-based inner network (top; $n = 2$ in this figure) and the MLP-based outer network that replaces ReLU with 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.

can be computed by networks of neurons, the finding that even single neurons can too highlights the possibility that individual neurons may be much more sophisticated than is often assumed in machine learning. 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 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.

2 Related work

Numerous recent studies have focused on developing novel activation functions, building on the simplicity and reliability of ReLU (Hahnloser et al., 2000; Nair and Hinton, 2010). 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 (He et al., 2015), ELU (Clevert et al., 2015), SELU (Klambauer et al., 2017), and GELU (Hendrycks and Gimpel, 2016) provide single-argument activation functions with a small number of parameters that are mostly fixed (or tuned through hyperparameter optimization). However, such hand-designed functional forms result in restricted expressivity. *Swish* (Ramachandran et al., 2017) 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 (Lin et al., 2013). 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) (Scarselli et al., 2008; Li et al., 2015; Kipf and

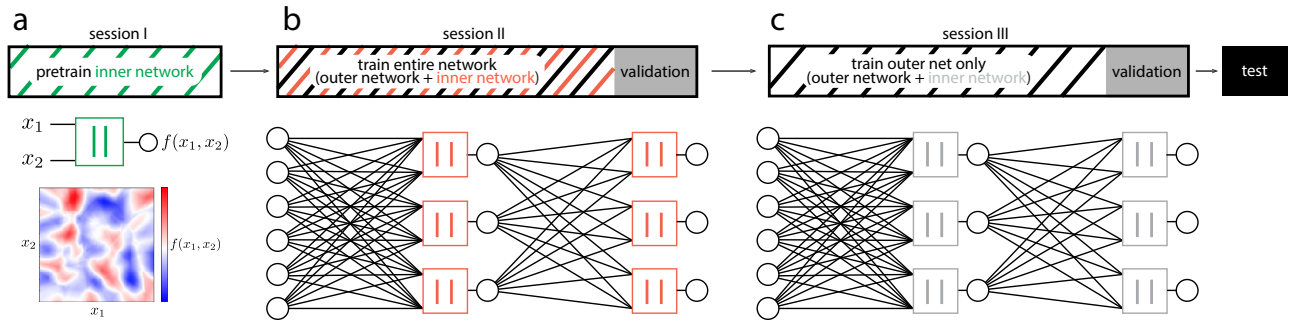

Figure 3: **Training procedure. (a)** Pretraining. Schematic of two-input argument inner network (green) trained to predict a smoothed random initial activation map (bottom). **(b)** Simultaneously training inner (red) and outer (black) networks. **(c)** Retraining outer network (black) with frozen inner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

Welling, 2016; Hamilton et al., 2017) and transformers (Vaswani et al., 2017; Jaegle et al., 2021). 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.

3 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 $f(x_1, ..., x_n)$ 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 essentially the same as 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 (Tolstikhin et al., 2021). Figure 2 summarizes how the inner network is incorporated into the outer network.

4 Experiments

4.1 Training procedure

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 that common initialization methods (Glorot and Bengio, 2010; He et al., 2016) do not generate spatial activations that are "random" enough to study the changes in functions over time. 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.

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.

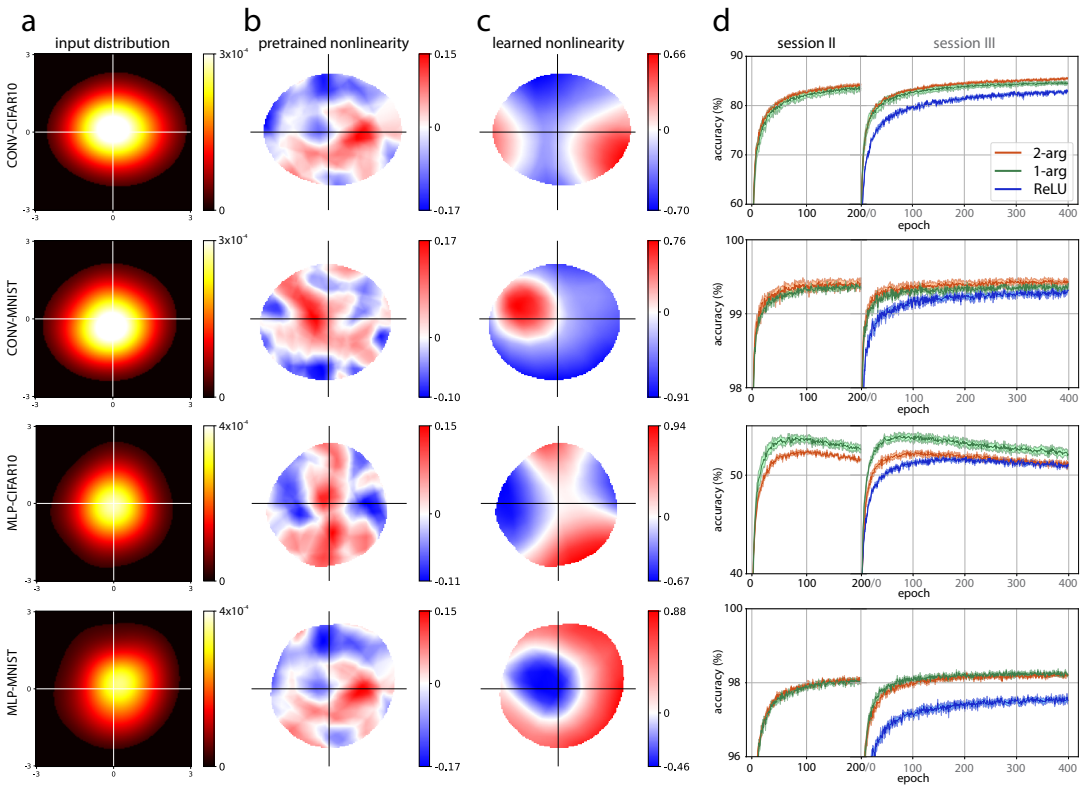

Figure 4: **Learned nonlinearities learn tasks faster.** Examples of **(a)** input distribution, **(b)** pretrained random initial nonlinearities, and **(c)** learned two-argument activation functions trained on two different data sets, CIFAR-10 and MNIST, within two different architecture types, a convolutional 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 activation model (red) and the baselines (blue: ReLU, green: 1-arg activation) in session II (200 epochs) and session III (400 epochs). Networks with these two-argument nonlinearities learn faster than others.

Aside from the MLPs or convolutional layers, the outer network uses other standard architectural components: layer normalization (Ba et al., 2016) (placed before inner networks) and dropout (Srivastava et al., 2014) (placed after each hidden/convolutional layer; $p = 0.5$). Our models are trained on the MNIST and CIFAR-10 datasets using ADAM (Kingma and Ba, 2014) 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 4.3).

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

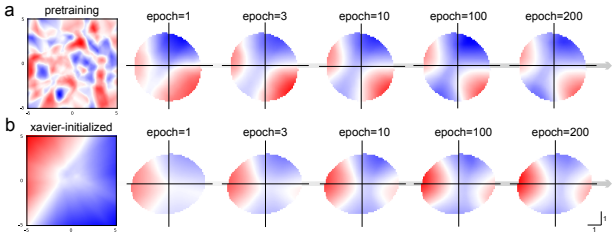

Figure 5: **Evolution of learned two-argument activation functions. (a)** Snapshot of random initial and learned nonlinear activation functions across development. **(b)** The same evolution of nonlinearity when it is Xavier-initialized.

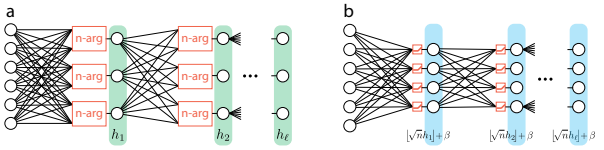

Figure 6: **Baseline architecture for parameter counts. (a)** MLP-based outer network that have L hidden layers with h_ℓ units (green) along with n-arg input nonlinearities (red). **(b)** Baseline model architecture with ReLU composed of L hidden layers with $\lfloor\sqrt{n}h_\ell\rfloor + \beta$ units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

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 the activation function emerges quite rapidly from pressures that arise early in the learning process.

4.2 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.

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, 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 dimension of input, output, and the number of 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 counts, so our baseline model (Figure 6b) has L layers, each comprising $\lfloor\sqrt{n}h_\ell\rfloor + \beta$ hidden units where β is a constant to approximate the parameter counts of the proposed model: $\lfloor\sqrt{n}h_\ell\rfloor \times \lfloor\sqrt{n}h_{\ell+1}\rfloor \approx nh_\ell h_{\ell+1}$. 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 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, Figure 4d 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.

4.3 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

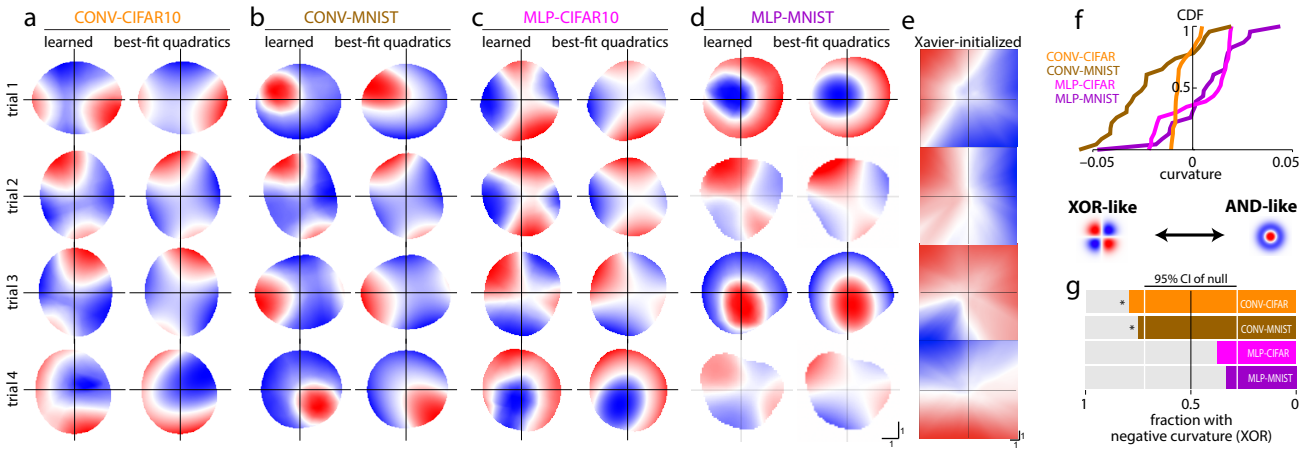

Figure 7: **Gating operations emerge naturally from learnable multi-argument nonlinear structures. (a-d) Left:** Examples of learned multi-argument activation functions trained on CIFAR-10 and MNIST, within two different architecture types, CNN and MLP. Each row is a different repetition of the learning 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).

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 (Glorot and Bengio, 2010) (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, $c_1 c_2 - c_3^2/4$ (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).

4.4 Spectral Analysis

To further compare the structure of learned nonlinearities to the structure of Xavier-initialized ones, we also performed a spectral analysis on both. We computed spectra using basis functions $\phi(\boldsymbol{x})$ appropriate for the symmetry and boundary conditions of the nonlinearities: we used Hermite-Bessel functions (Victor et al., 2006) for the 2-argument functions, and solid harmonics for the 3-argument 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 nonlineari-

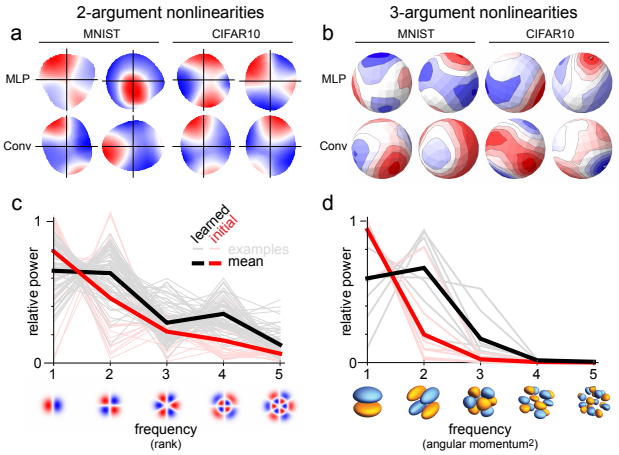

Figure 8: **Spectral Analysis.** Nonlinearities for various architectures and tasks for **(a)** two-argument and **(b)** three-argument inner networks. **(c–d)** Power spectra for these learned functions (black curves) reveal larger power at $\ell = 2$ than spectra for Xavier-initialized inner networks (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.

ties 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.

4.5 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 (Hendrycks and Dietterich, 2019) and parameter-free AutoAttack (Croce and Hein, 2020b).

Robustness against common image corruptions CIFAR-10-C was designed to measure the robustness

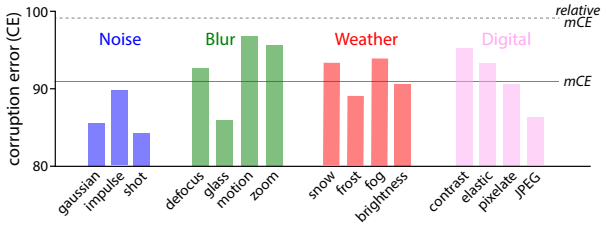

Figure 9: **Robustness of two-argument nonlinearities against common image corruptions.** Corruption error (CE; bars), mCE (black solid line), and relative mCE (black dashed line) of different corruptions on CIFAR-10-C and Conv-based outer networks. The mCE is the mean corruption error of the corruptions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

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. $\mathrm{CE}_c^{\text{2-arg}} = \left(\sum_{s=1}^5 E_{s,c}^{\text{2-arg}}\right) \Big/ \left(\sum_{s=1}^5 E_{s,c}^{\text{ReLU}}\right)$. The mean corruption error is then obtained by averaging over the corruption types: $\mathrm{mCE} = \langle \mathrm{CE}_c^{\text{2-arg}} \rangle_c$. We also compute a relative mCE score by subtracting the clean classification error of the classifiers from the corruption errors: Relative $\mathrm{CE}_c^{\text{2-arg}} = \left(\sum_{s=1}^5 E_{s,c}^{\text{2-arg}} - E_{\text{clean}}^{\text{2-arg}}\right) \Big/ \left(\sum_{s=1}^5 E_{s,c}^{\text{ReLU}} - E_{\text{clean}}^{\text{ReLU}}\right)$ and then averaging over different corruption types as before results in the relative $\mathrm{mCE} = \langle \text{relative } \mathrm{CE}_c^{\text{2-arg}} \rangle_c$. 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.

Adversarial robustness We next consider both

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

black-box and white-box attacks to measure the robustness of the model against adversarial perturbations. We use the recently introduced AutoAttack (Croce and Hein, 2020b) combining two parameter-free versions of Projected Gradient Descent (PGD) (Madry et al., 2017) algorithm with two existing complementary Fast Adaptive Boundary (FAB) (Croce and Hein, 2020a) and Square (Andriushchenko et al., 2020) 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.

5 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 (Hochreiter and Schmidhuber, 1997), GRUs (Chung et al., 2014), and transformers (Vaswani et al., 2017). 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 (Gidon et al., 2020). 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 4.3, 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 (Williams et al., 2019; Sahs et al., 2020) 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 Lin et al. (2013), and to see whether different tasks incentivize different computations.

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 (Douglas and Martin, 1991; Shepherd, 2004), and more intricate dendritic substructures (Poirazi and Mel, 2001; Poirazi

et al., 2003). Such biologically-inspired additions to neural network architectures could provide inductive biases closer to the inductive biases in biological brains (Sinz et al., 2019; Litwin-Kumar and Turaga, 2019).

References

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. (2020). Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Barlow, H. B. et al. (1961). Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01).

Beniaguev, D., Segev, I., and London, M. (2021). Single cortical neurons as deep artificial neural networks. *Neuron*, 109(17):2727–2739.

Branco, T., Clark, B. A., and Häusser, M. (2010). Dendritic discrimination of temporal input sequences in cortical neurons. *Science*, 329(5999):1671–1675.

Carandini, M. and Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51–62.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Croce, F. and Hein, M. (2020a). Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pages 2196–2205. PMLR.

Croce, F. and Hein, M. (2020b). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR.

Douglas, R. J. and Martin, K. (1991). A functional microcircuit for cat visual cortex. *The Journal of physiology*, 440(1):735–769.

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

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heeger, D. J. (1992). Normalization of cell responses in cat striate cortex. *Visual neuroscience*, 9(2):181–197.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., and Carreira, J. (2021). Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*.

Jones, I. S. and Kording, K. P. (2021). Might a single neuron solve interesting machine learning problems through successive computations on its dendritic tree? *Neural Computation*, 33(6):1554–1571.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981.

Larkum, M. E., Zhu, J. J., and Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving

at different cortical layers. *Nature*, 398(6725):338–341.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Litwin-Kumar, A. and Turaga, S. C. (2019). Constraining computational models using electron microscopy wiring diagrams. *Current opinion in neurobiology*, 58:94–100.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Minsky, M. and Papert, S. (1969). Perceptrons.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*.

Poirazi, P., Brannon, T., and Mel, B. W. (2003). Pyramidal neuron as two-layer neural network. *Neuron*, 37(6):989–999.

Poirazi, P. and Mel, B. W. (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron*, 29(3):779–796.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Rieke, F. and Warland, D. (1999). *Spikes: exploring the neural code*. MIT press.

Sahs, J., Pyle, R., Damaraju, A., Caro, J. O., Tavaslioglu, O., Lu, A., and Patel, A. (2020). Shallow univariate relu networks as splines: initialization, loss surface, hessian, & gradient flow dynamics. *arXiv preprint arXiv:2008.01772*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Shepherd, G. M. (2004). *The synaptic organization of the brain*. Oxford university press.

Sinz, F. H., Pitkow, X., Reimer, J., Bethge, M., and Tolias, A. S. (2019). Engineering a less artificial intelligence. *Neuron*, 103(6):967–979.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Victor, J. D., Mechler, F., Repucci, M. A., Purpura, K. P., and Sharpee, T. (2006). Responses of v1 neurons to two-dimensional hermite functions. *Journal of neurophysiology*, 95(1):379–400.

Williams, F., Trager, M., Silva, C., Panozzo, D., Zorin, D., and Bruna, J. (2019). Gradient dynamics of shallow univariate relu networks. *arXiv preprint arXiv:1906.07842*.