Logical Activation Functions: Logit-space equivalents of Probabilistic Boolean Operators

Scott C. Lowe^{1,2,*}, **Robert Earle**^{1,2}, **Jason d'Eon**^{1,2}, **Thomas Trappenberg**^{1}, **Sageev Oore**^{1,2}

^{1}Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia
Canada

^{2}Vector Institute for Artificial Intelligence
Toronto, Ontario
Canada

*Correspondence: scottclowe@gmail.com

Abstract

The choice of activation functions and their motivation is a long-standing issue within the neural network community. Neuronal representations within artificial neural networks are commonly understood as logits, representing the log-odds score of presence of features within the stimulus. We derive logit-space operators equivalent to probabilistic Boolean logic-gates AND, OR, and XNOR for independent probabilities. Such theories are important to formalize more complex dendritic operations in real neurons, and these operations can be used as activation functions within a neural network, introducing probabilistic Boolean-logic as the core operation of the neural network. Since these functions involve taking multiple exponents and logarithms, they are computationally expensive and not well suited to be directly used within neural networks. Consequently, we construct efficient approximations named AND_{AIL} (the AND operator Approximate for Independent Logits), OR_{AIL}, and XNOR_{AIL}, which utilize only comparison and addition operations, have well-behaved gradients, and can be deployed as activation functions in neural networks. Like MaxOut, AND_{AIL} and OR_{AIL} are generalizations of ReLU to two-dimensions. While our primary aim is to formalize dendritic computations within a logit-space probabilistic-Boolean framework, we deploy these new activation functions, both in isolation and in conjunction to demonstrate their effectiveness on a variety of tasks including tabular classification, image classification, transfer learning, abstract reasoning, and compositional zero-shot learning.

1 Introduction

Non-linear activation functions are essential in artificial neural networks (ANNs) to form higher-order representations, since otherwise the network would be degeneratively equivalent to a single linear layer. Most activation functions represent a simple non-linearity despite evidence of much more complex non-linear integration and computations in dendrites (Hentschel et al., 2004; London & Häusser, 2005; Payeur et al., 2019). For example, Gidon et al. (2020) recently demonstrated that a single biological neuron can compute the XOR of its inputs, a property long known to be lacking from artificial neurons (Minsky & Papert, 1969). While simple activation functions work in ANNs in the sense that more complex operations can be formed from the combination of several layers of neurons, understanding the function and impact of advanced dendritic operations in networks is important. In this work, we add some of this behaviour to neural activations, corresponding to shifting some of the network's complexity from its global structure to the neural level. There is far more complexity in biological neurons than in the abstractions that we consider here, but we make a step in the direction of using more complex neurons in ANNs. We also develop a theoretical underpinning for higher-order activation functions (e.g. like MaxOut, Goodfellow et al., 2013) in a probabilistic framework. We hypothesize that such architectures will be more parameter-efficient in situations where their assumptions hold.

Neuronal representations within ANNs are commonly understood as logits, representing the log-odds score of presence (versus absence) of features within the stimulus. From a Bayesian perspective, a ReLU-like operation corresponds to the removal of all evidence for the lack of a feature. Under the logit interpretation of ANN potentiations, this seems unreasonable. This can be seen in some of the ways we interact with neural networks: we must apply batch-norm before activations and not after them; when doing transfer learning from an embedding space we must use pre-activation potentiations instead of activations. Can ANNs do better if we design an architecture which treats potentiations as logits?

We thus derive logit-space operators equivalent to probabilistic Boolean logic-gates AND, OR, and XNOR for independent probabilities. Networks constructed in this way can be interpreted as **performing logical operations** using **point-estimates of probabilities**, in a similar manner to a Bayesian network. This brings operations from the symbolism framework of AI (which is more similar to deliberative thinking, or System 2 of Kahneman, 2011) into the connectionist framework of ANNs (which is more like instinctive, System 1, thinking). We also construct computationally feasible approximations to these functions with well-behaved gradients. These new activation functions, which are generalizations of ReLU to two-dimensions, are then applied on benchmark datasets to demonstrate their effectiveness on a diverse range of tasks including image classification, transfer learning, abstract reasoning, and compositional zero-shot learning. The new principled approach we present introduces new ways to redistribute computation from the network into the neuronal mechanisms, and build more parameter-efficient models. We demonstrate the effectiveness of activation functions based on these ideas, and expect future work to build on this.

2 Background

Early artificial neural networks featured either logistic-sigmoid or tanh as their activation function, motivated by the idea that each layer of the network is building another layer of abstraction of the stimulus space from the last layer. Each neuron in a layer identifies whether certain properties or features are present within the stimulus, and the pre-activation (potentiation) value of the neuron indicates a score or logit for the presence of that feature. The sigmoid function, $\sigma(x) = {}^1\!/\!_{(1+e^{-x})}$, was hence a natural choice of activation function, since as with logistic regression, this will convert the logits of features into probabilities. There is evidence that this interpretation has merit. Previous work has been done to identify which features neurons are tuned to. Examples include LSTM neurons tracking quotation marks, line length, and brackets (Karpathy et al., 2015) and sentiment (Radford et al., 2017); projecting features back to the input space to view them (Olah et al., 2017); and interpretable combinations of neural activities (Olah et al., 2020).

Sigmoidal activation functions are no longer commonly used within machine learning between layers of representations, though sigmoid is still widely used for gating operations which scale the magnitude of other features in an attention-like manner. The primary disadvantage of the sigmoid activation function is its vanishing gradient — as the potentiation rises, activity converges to a plateau, and hence the gradient goes to zero. This prevents feedback information propagating back through the network from the loss function to the early layers of the network, which consequently prevents it from learning to complete the task.

The Rectified Linear Unit activation function (Fukushima, 1980; Jarrett et al., 2009; Nair & Hinton, 2010), $\mathrm{ReLU}(x) = \max(0, x)$, does not have this problem, since in its non-zero regime it has a gradient of 1. Another advantage of ReLU is it has very low computational demands. Since it is both effective and efficient, it has proven to be a highly popular choice of activation function. The chief drawback to ReLU is that it has no sensitivity to changes across half of its input domain, which prevents updates on stimuli which trigger its "off" state and can even lead to neuronal death[1]. Variants of ReLU have emerged, aiming to smooth out its transition between domains and provide

[1]Though this problem is very rare when using BatchNorm to stabilize feature distributions.

a gradient in its inactive regime. These include ELU (Clevert et al., 2016), CELU (Barron, 2017), SELU (Klambauer et al., 2017), GELU (Hendrycks & Gimpel, 2020), SiLU (Elfwing et al., 2018; Ramachandran et al., 2017), and Mish (Misra, 2019). However, all these activation functions still bear the general shape of ReLU and truncate negative logits.

Fuzzy logic operators are generalizations of Boolean logic operations to continuous variables, using rules similar to applying logical operators in probability space. Prior work has explored networks of fuzzy logic operations, including some which use an activation function constituting a learnable interpolation between fuzzy logic operators (Godfrey & Gashler, 2017). The activation functions we introduce here are motivated similarly to fuzzy logic operators, but designed to operate in logit space instead of in probability space, which better reflects the behaviour and space of pre-activation units.

Bayesian neural networks (BNNs) are probabilistic models which can represent the uncertainty in their model parameters, and hence uncertainty in their outputs. This differs from standard ANNs which use only a single point-estimate of each model parameter, and hence also in their neural activations. By making their priors explicit and modelling their uncertainty, Bayesian networks can be better calibrated and less vulnerable to overfitting than ANNs. However, BNNs are more challenging to train, and cannot reasonably be scaled up to the deep architectures which are possible with standard ANNs and necessary in order to learn a sufficiently complex model to solve highly complex tasks.

In this work we contribute to the theoretical underpinning of neural activation functions by developing activation functions based on the principle that neurons encode logits — scores that represent the presence of features in the log-odds space. In § 3 we derive and define these functions in detail for different logical operators, and then consider their performance on numerous task types including parity (§ 4.1), image classification (§ 4.3 and § 4.4), transfer learning (§ 4.5), abstract reasoning (Appendix § A.18), soft-rule guided classification as exemplified by the Bach chorale dataset (§ 4.2), and compositional zero-shot learning (Appendix § A.19). These tasks were selected to (1) survey the performance of the new activations on existing benchmark tasks, and (2) evaluate their performance on tasks which we suspect in particular may require logical reasoning and hence benefit from activation functions which apply these logical operations to logits.

3 Derivation

Manipulation of probabilities in logit-space is known to be more efficient for many calculations. For instance, the log-odds form of Bayes' Rule (Appendix Eq. 9) states that the posterior logit equals the prior logit plus the log of the likelihood ratio for the new evidence (the log of the Bayes factor). Thus, working in logit-space allows us to perform Bayesian updates on many sources of evidence simultaneously, merely by summing together the log-likelihood ratios for the evidence (see Appendix § A.3). A weighted sum may be used if the amount of credence given to the sources differs — and this is precisely the operation performed by a linear layer in a neural network.

When considering sets of probabilities, a natural operation is to measure the joint probability of two events both occurring— the AND operation. Suppose our input space is $x \in [0, 1]^2$, and the goal is to output $y > 0$ if $x_i = 1 \, \forall \, i$, and $y < 0$ otherwise, using model with a weight vector w and bias term b, such that $y = w^T x + b$. This can be trivially solved with the weight matrix $w = [1, 1]$ and bias term $b = -1.5$. However, since this is only a linear separator, the solution can not generalize to the case $y > 0$ iff $x_i > 0 \, \forall \, i$. Similarly, let us consider how the OR function is solved with a linear layer. Our goal is to output $y > 0$ if $\exists \, x_i = 1$, and $y < 0$ otherwise. The binary case can be trivially solved with the weight matrix $w = [1, 1]$ and bias term $b = -0.5$. The difference between the solution for OR and the solution for AND is only an offset to our bias term. In each case, if the input space is expanded beyond binary to $\mathbb{R}^2$, the output can be violated by changing only one of the arguments.

3.1 AND

Suppose we are given x and y as the logits for the presence (vs absence) of two events, X and Y. These logits have equivalent probability values, which can be obtained using the sigmoid function, $\sigma(u) = (1 + e^{-u})^{-1}$. Let us assume that the events X and Y are independent of each other. In this case, the probability of both events occurring (the joint probability) is $\mathrm{P}(X, Y) = \mathrm{P}(X \wedge Y) = \mathrm{P}(X) \; \mathrm{P}(Y) = \sigma(x) \, \sigma(y)$. However, we wish to remain in logit-space, and must determine the logit of the joint probability, $\operatorname{logit}(\mathrm{P}(X, Y)$. This is given by

$$\begin{aligned} \mathrm{AND}_{\mathrm{IL}} &:= \operatorname{logit}(\mathrm{P}(X \wedge Y)_{x \perp\!\!\!\perp y}) = \log\left(\frac{p}{1-p}\right), \text{ where } p = \sigma(x)\,\sigma(y), \\ &= \log\left(\frac{\sigma(x)\,\sigma(y)}{1-\sigma(x)\,\sigma(y)}\right), \end{aligned} \tag{1}$$

which we coin as $\mathrm{AND}_{\mathrm{IL}}$, the AND operator for independent logits (IL). This 2d function is illustrated as a contour plot (Fig. 1, left panel). Across the plane, the order of magnitude of the output is the same as at least one of the two inputs, scaling approximately linearly.

The approximately linear behaviour of the function is suitable for use as an activation function (no vanishing gradient), however taking exponents and logs scales poorly from a computational perspective. Hence, we developed a computationally efficient approximation as follows. Observe that we can loosely approximate $\mathrm{AND}_{\mathrm{IL}}$ with the minimum function (Fig. 1, right panel). This is equivalent to assuming the probability of **both** X and Y being true equals the probability of the **least likely** of X and Y being true — a naïve approximation which holds well in three quadrants of the plane, but overestimates the probability when both X and Y are unlikely. In this quadrant, when both X and Y are both unlikely, a better approximation for $\mathrm{AND}_{\mathrm{IL}}$ is the sum of their logits.

We thus propose $\mathrm{AND}_{\mathrm{AIL}}$, a linear-approximate AND function for independent logits (AIL, i.e. approximate IL).

$$\mathrm{AND}_{\mathrm{AIL}}(x, y) := \begin{cases} x + y, & x < 0,\ y < 0 \\ \min(x, y), & \text{otherwise} \end{cases} \tag{2}$$

As shown in Fig. 1 (left, middle), we observe that their output values and shape are very similar.

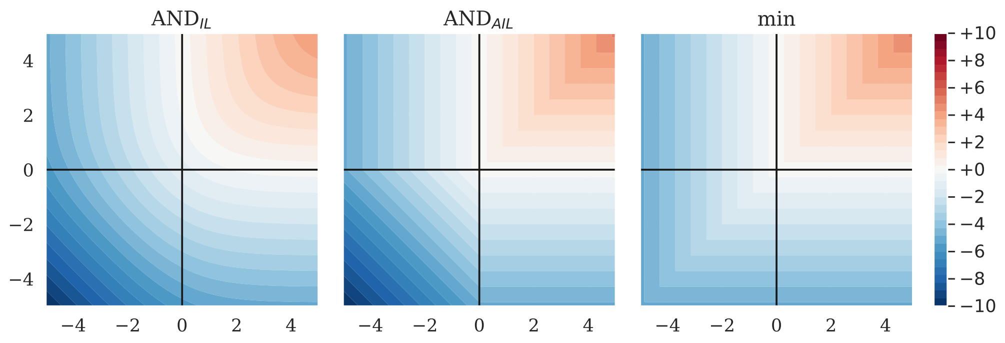

Figure 1: Heatmap comparing the outputs for the exact logit-space probabilistic-and function for independent logits, $\mathrm{AND}_{\mathrm{IL}}(x, y)$; our constructed approximation, $\mathrm{AND}_{\mathrm{AIL}}(x, y)$; and $\max(x, y)$.

3.2 OR

Similarly, we can construct the logit-space OR function, for independent logits. For a pair of logits x and y, the probability that either of the corresponding events is true is given by $p = 1 - \sigma(-x)\,\sigma(-y)$. This can be converted into a logit as

$$\mathrm{OR}_{\mathrm{IL}}(x, y) := \operatorname{logit}(\mathrm{P}(X \vee Y)_{x \perp\!\!\!\perp y}) = \log\left(\frac{p}{1-p}\right), \text{ with } p = 1 - \sigma(-x)\,\sigma(-y) \tag{3}$$

which can be roughly approximated by the max function. This is equivalent to setting the probability of **either** of event X or Y occurring to be equal to the probability of the **most likely** event. This underestimates the upper-right quadrant (below), which we can approximate better as the sum of the input logits, yielding

$$\mathrm{OR}_{\mathrm{AIL}}(x, y) := \begin{cases} x + y, & x > 0,\ y > 0 \\ \max(x, y), & \text{otherwise} \end{cases} \tag{4}$$

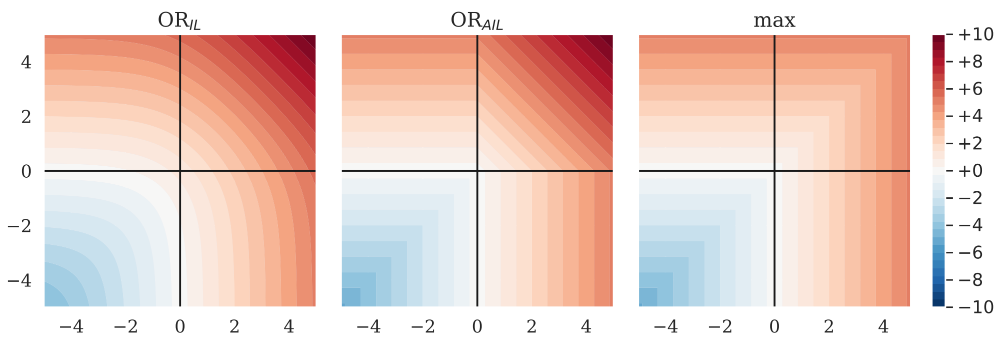

Figure 2: Comparison of the exact logit-space probabilistic-or function for independent logits, $\mathrm{OR}_{\mathrm{IL}}(x, y)$; our constructed approximation, $\mathrm{OR}_{\mathrm{AIL}}(x, y)$; and $\max(x, y)$.

3.3 XNOR

We also consider the construction of a logit-space XNOR operator. This is the probability that X and Y occur either together, or not at all, given by

$$\mathrm{XNOR}_{\mathrm{IL}}(x, y) := \mathrm{logit}(\mathrm{P}(X \bar{\oplus} Y)_{x \perp\!\!\!\perp y}) = \log\left(\frac{p}{1-p}\right), \tag{5}$$

where $p = \sigma(x)\,\sigma(y) + \sigma(-x)\,\sigma(-y)$. We can approximate this with

$$\mathrm{XNOR}_{\mathrm{AIL}}(x, y) := \mathrm{sgn}(xy) \min(|x|, |y|), \tag{6}$$

which focuses on the logit of the feature **most likely** to **flip** the expected **parity** (see Fig. 3).

We could use other approximations, such as the sign-preserving geometric mean,

$$\mathrm{SignedGeomean}(x, y) := \mathrm{sgn}(xy)\sqrt{|xy|}, \tag{7}$$

but this matches $\mathrm{XNOR}_{\mathrm{IL}}$ less closely, and has a divergent gradient along both $x = 0$ and $y = 0$.

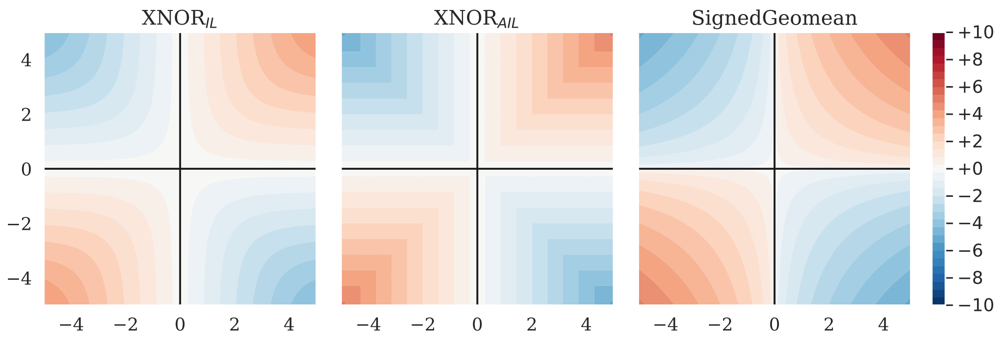

Figure 3: Comparison of the exact logit-space probabilistic-xnor function for independent logits, $\mathrm{XNOR}_{\mathrm{IL}}(x, y)$; our constructed approximation, $\mathrm{XNOR}_{\mathrm{AIL}}(x, y)$; and $\mathrm{SignedGeomean}(x, y)$.

3.4 Discussion

By working via probabilities, and assuming inputs encode independent events, we have derived logit-space equivalents of the Boolean logic functions, AND, OR, and XNOR. Since these are computationally demanding to compute repeatedly within a neural network, we have constructed approximations of them: $\mathrm{AND}_{\mathrm{AIL}}$, $\mathrm{OR}_{\mathrm{AIL}}$, and $\mathrm{XNOR}_{\mathrm{AIL}}$. Like ReLU, these involve only comparison, addition, and multiplication operations which are cheap to perform. In fact, $\mathrm{AND}_{\mathrm{AIL}}$ and $\mathrm{OR}_{\mathrm{AIL}}$ are a generalization of ReLU to an extra dimension, since $\mathrm{OR}_{\mathrm{AIL}}(x, y = 0) = max(x, 0)$.

The majority of activation functions are one-dimensional, $f : \mathbb{R} \to \mathbb{R}$. In contrast to this, our proposed activation functions are all two-dimensional, $f : \mathbb{R}^2 \to \mathbb{R}$. They must be applied to pairs of features from the embedding space, and will reduce the dimensionality of the input space by a factor of 2. This behaviour is the same as seen in MaxOut networks (Goodfellow et al., 2013) which use $\max$ as their activation function; $\mathrm{MaxOut}(x, y; k) := \max(x, y)$. Similar to MaxOut, our activation functions could be generalized to higher dimensional inputs, $f : \mathbb{R}^k \to \mathbb{R}$, by considering the behaviour of the logit-space AND, OR, XNOR operations with regard to more inputs. For simplicity, we restrict this work to consider only $k = 2$, but note these activation functions also generalize to higher dimensions.

3.5 Ensembling

By using multiple logit-Boolean activation functions *simultaneously* alongside each other, we permit the network multiple options of how to relate features together. When combining the activation functions, we considered two strategies (illustrated in Appendix §A.6).

In the partition (p) strategy, we split the n_c dimensional pre-activation embedding equally into m partitions, apply different activation functions on each partition, and concatenate the results together. Using AIL activation functions under this strategy, the output dimension will always be half that of the input, as it is for each AIL activation function individually. In the duplication (d) strategy, we apply m different activation functions in parallel to the same n_c elements. The output is consequently larger, with dimension $m\, n_c$. If desired, we can counteract the $2 \to 1$ reduction of AIL activation functions by using two of them together under this strategy. A negative weight in a network is equivalent to the logit-NOT operator. Hence with sufficient width and depth, a multi-layer network using only the $\mathrm{OR}_{\mathrm{IL}}$ activation function should be able to replicate any probabilistic logic circuit.

Utilizing $\mathrm{AND}_{\mathrm{AIL}}$, $\mathrm{OR}_{\mathrm{AIL}}$ and $\mathrm{XNOR}_{\mathrm{AIL}}$ simultaneously allows our networks to access logit-space equivalents of 12 of the 16 Boolean logical operations with only a single sign inversion (in either one of the inputs or the output). Including the bias term and skip connections, the network has easy access to logit-space equivalents of all 16 Boolean logical operations.

We hypothesised that training a network with all three of our activation functions in an ensemble in this manner could yield better results since the network would not need to expend layers having to combine $\mathrm{OR}_{\mathrm{IL}}$ operations together to yield other Boolean operations.

3.6 Normalization

Our AIL activation functions are close approximations to exact AND, OR, and XNOR operations in logit-space. However, when deploying non-linearities within a neural network, it is important that the activation functions have a gain of 1 in order to improve stability during training (Klambauer et al., 2017), a property the AIL activations do not possess. We constructed normalized variants of the exact and approximate logit operators, dubbed NIL and NAIL respectively, by subtracting the expected mean and dividing by the expected standard deviation, assuming the operands are sampled independently from the standard normal distribution, $\mathcal{N}(0, 1)$. For more details, see Appendix §A.7.

4 Experiments

We evaluated the performance of our AIL activations, both individually and together in an ensemble, on a range of benchmark tasks. Since $\mathrm{AND}_{\mathrm{AIL}}$ and $\mathrm{OR}_{\mathrm{AIL}}$ are equivalent when the sign of operands and outputs can be freely chosen, we only show results for $\mathrm{OR}_{\mathrm{AIL}}$. We compared against three primary baselines: (1) ReLU, (2) $\max(x, y) = \mathrm{MaxOut}([x, y]; k = 2)$, and (3) the concatenation of $\max(x, y)$ and $\min(x, y)$, denoted $\{\mathrm{Max}, \mathrm{Min}\ (\mathrm{d})\}$. The $\{\mathrm{Max}, \mathrm{Min}\ (\mathrm{d})\}$ ensemble is equivalent to $\mathrm{GroupSort}$ with a group size of 2 (Chernodub & Nowicki, 2017; Anil et al., 2019), sometimes referred to as the MaxMin operator; it is comparable to the concatenation of $\mathrm{OR}_{\mathrm{AIL}}$ and $\mathrm{AND}_{\mathrm{AIL}}$ under our duplication strategy.

Source code for our experiments can be found at `https://github.com/DalhousieAI/logical_activation_experiments`. Our python package which provides an implementation of these the activation functions is available at `https://github.com/DalhousieAI/pytorch-logit-logic`, which is also available as PyPI package `pytorch-logit-logic`.

4.1 Parity

In a simple initial experiment, we constructed a synthetic dataset whose labels could be derived directly by stacking the logical operation XNOR. Each sample had four input logits, and target value equal to the parity of the number of positive inputs. A very small model using XNOR with two hidden layers (of 4, then 2 neurons) should be capable of perfect classification accuracy on this dataset with a sparse weight matrix by learning to nest pairwise binary relationships. We trained such an MLP with either ReLU or $\mathrm{XNOR}_{\mathrm{AIL}}$ activations.

The MLP with $\mathrm{XNOR}_{\mathrm{AIL}}$ activation learned a sparse weight matrix able to perfectly classify any input combination, shown in Fig. 4. In comparison, with ReLU the network was only able to produce 60% classification accuracy. The accuracy with ReLU was improved by increasing the MLP width/depth, but this did not result in a sparse weight matrix. This experiment demonstrates that $\mathrm{XNOR}_{\mathrm{AIL}}$ can be utilized by a network to find the simplest relationship between inputs that satisfies the objective. For additional results, see Appendix § A.11.

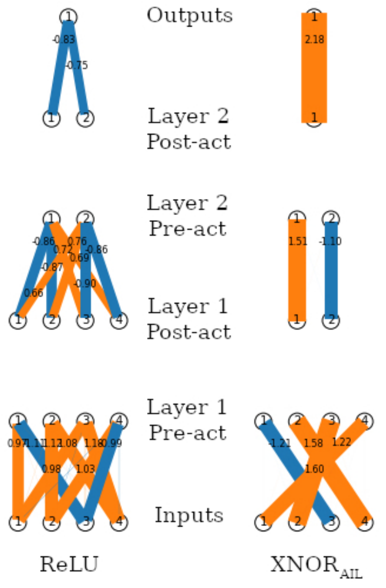

Figure 4: Visualization of weight matrices learnt by two-layer MLPs on a binary classification task, where the target output is the parity of the inputs. Line widths indicate weight magnitudes (orange: +ve, blue: -ve). MLP with ReLU: 60% accuracy; $\mathrm{XNOR}_{\mathrm{AIL}}$: 100% accuracy.

4.2 MLP on Bach Chorales and Logit Independence

The Bach Chorale dataset (Boulanger-Lewandowski et al., 2012) consists of 382 chorales composed by JS Bach, each ∼12 measures long, totalling ∼83 000 notes. Represented as discrete sequences of tokens, it has served as a benchmark for music processing for decades, from heuristic methods to HMMs, RNNs, and CNNs (Mozer, 1990; Hild et al., 1991; Allan & Williams, 2005; Liang, 2016; Hadjeres et al., 2017; Huang et al., 2019). The chorales are comprised of 4 voices (melodic lines) whose behaviour is guided by soft musical rules depending on the prior movement of that and other voices, e.g. "two voices a fifth apart ought not to move in parallel with one another". We tasked 2-layer MLPs with determining whether a short four-part musical excerpt is taken from a Bach chorale. Negative examples were created by stochastically corrupting chorale excerpts (see Appendix § A.13). We found $\{\mathrm{OR}, \mathrm{AND}, \mathrm{XNOR}_{\mathrm{AIL}}$ (d)$\}$ had highest accuracy, but the results were not statistically significant ($p < 0.1$ between best and worst, two-tailed Student's t-test, 10 random inits; see Appendix § A.10).

Additionally, we investigated the independence between logits in the trained pre-activation embeddings. We expect that an MLP which is optimally engaging its neurons would maintain independence between features in order to maximize information. We measured correlations between adjacent pre-activations (paired operands for the logical activation functions), and also between non-adjacent pairs of pre-activations. Our results indicate the network learns features which are independent when they are not passed to the same 2D activation, and *anti-correlated* features when they are. For more details, see Appendix § A.14.

4.3 CNN and MLP on MNIST

We trained 2-layer MLP and 6-layer CNN models on MNIST with ADAM (Kingma & Ba, 2015), 1-cycle schedule (Smith & Topin, 2017; Smith, 2018), and using hyperparameters tuned through a random search against a validation set comprised of the last 10k images of the training partition.

The MLP used two hidden layers, the widths of which were varied together to evaluate the performance for a range of model sizes. The CNN used six layers of 3x3 convolution layers, with 2x2 max pooling (stride 2) after every other conv layer, followed by flattening and three MLP layers. The layer widths were scaled up to explore a range of model sizes (see Appendix § A.15 for more details).

For the MLP, $\mathrm{XNOR}_{\mathrm{AIL}}$ performed best along with $\mathrm{SignedGeomean}$ ($p < 0.1$, two-tailed Student's t-test), ahead of all other activations ($p < 0.01$ for each; Fig. 5 left panel) when considering the best

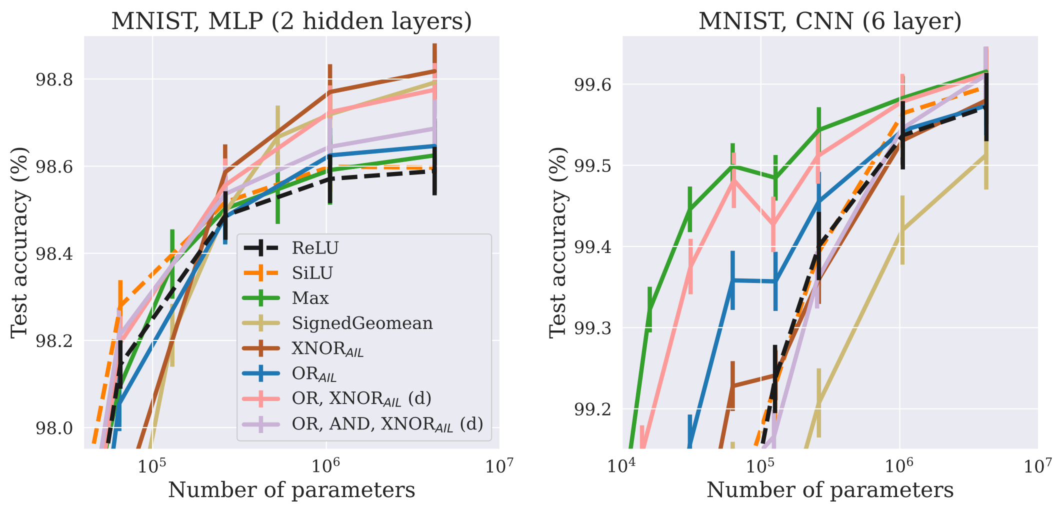

Figure 5: We trained CNN on MNIST, MLP on flattened-MNIST, using ADAM (1-cycle, 10 ep), hyperparams determined by random search. Mean (bars: std dev) of $n = 40$ weight inits.

performance across all widths (see Appendix § A.10 for methodology). However when the width is reduced below 2×10^5 there is a transition and XNOR-shaped activations perform worst. We hypothesize this may be because smaller widths embeddings are over-saturated and have individual units corresponding to multiple features, whilst XNOR activations may require single-feature units to perform best.

With the CNN, five activation configurations ($\{\mathrm{OR}, \mathrm{AND}, \mathrm{XNOR}_{\mathrm{AIL}}$ (p)$\}$, $\{\mathrm{OR}, \mathrm{XNOR}_{\mathrm{AIL}}$ (d/p)$\}$, Max, and SiLU) performed best ($p < 0.05$ for other activations, two-tailed Student's t-test; Fig. 5, right panel; Appendix § A.10). CNNs which used $\mathrm{OR}_{\mathrm{AIL}}$ or Max (alone or in an ensemble) maintained high performance with an order of magnitude fewer parameters (3×10^4) than others (3×10^5 params).

4.4 ResNet50 on CIFAR-10/100

We explored the impact of our activation functions on deep networks by deploying them in a pre-activation ResNet50 model (He et al., 2016a,b). We exchanged all ReLU activations in the network to a candidate activation while maintaining the size of the pass-through embedding. We experimented with changing the width of the network, scaling up the embedding space and all hidden layers by a common factor, w. The network was trained on CIFAR-10/-100 for 100 epochs using ADAM (Kingma & Ba, 2015), 1-cycle (Smith, 2018; Smith & Topin, 2017). See Appendix § A.16 for further details.

For both CIFAR-10 and -100, SiLU, $\mathrm{OR}_{\mathrm{AIL}}$, and Max outperform ReLU across a wide range of width values (see Fig. 6). These three activation functions hold up their performance best as the number of parameters is reduced. Since SiLU was discovered by a search of $1 \to 1$ activation functions for this type of architecture and task (Ramachandran et al., 2017), we expected it to perform well in this setting. Yet, we find $\mathrm{OR}_{\mathrm{AIL}}$ and Max perform better than ReLU and comparable with SiLU generally. Meanwhile, other AIL activations perform similarly to ReLU when the width is thin, and slightly worse than ReLU when the width is wide. When used on its own and not part of an ensemble, the $\mathrm{XNOR}_{\mathrm{AIL}}$ activation function performed poorly (off the bottom of the chart), indicating it is not suited for this task.

4.5 Transfer learning

We considered transfer learning on several image classification datasets. We used a ResNet18 model (He et al., 2016a) pretrained on ImageNet-1k. The weights were frozen (not fine-tuned) and used to generate embeddings of samples from other image datasets. We trained a two-layer MLP to classify images from these embeddings using various activation functions. For a compre-

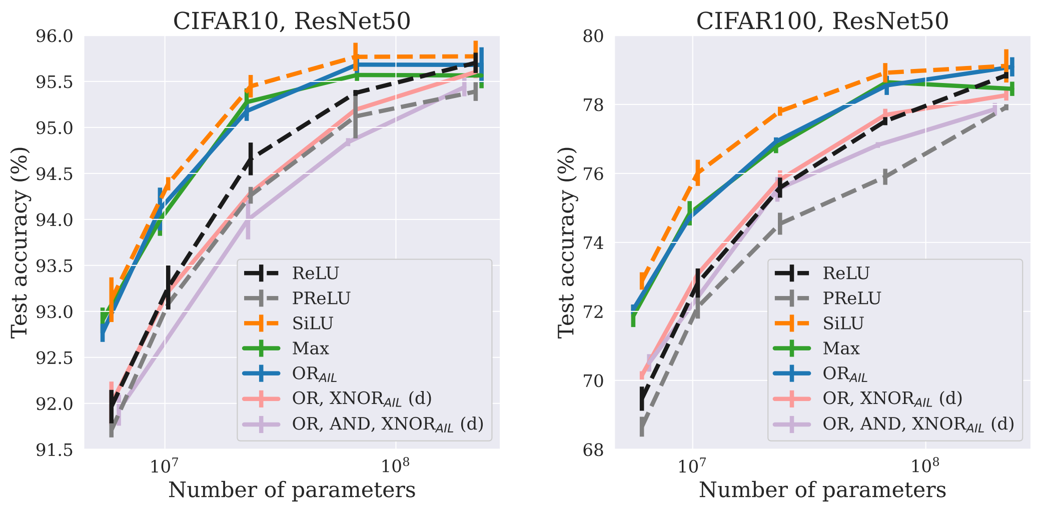

Figure 6: ResNet50 on CIFAR-10/100, varying the activation function used through the network. The width was varied to explore a range of network sizes (see text). Trained for 100 ep. with ADAM, using hyperparams as determined by random search on CIFAR-100 with width factor $w = 2$. Mean (bars: std dev) of $n = 4$ weight inits.

hensive set of baselines, we compared against every activation function built into PyTorch 1.10 (see Appendix § A.17). To make the number of parameters similar, we used a width of 512 for activation functions with $1 \to 1$ mapping (e.g. ReLU), a width of 650 for activation functions with a $2 \to 1$ mapping (e.g. Max, $\mathrm{OR}_{\mathrm{AIL}}$), and a width of 438 for $\{\mathrm{OR}, \mathrm{AND}, \mathrm{XNOR}_{\mathrm{AIL}}\ (\mathrm{d})\}$. See Appendix § A.17 for more details.

Our results are shown in Table 1. We found that all our (N)AIL activation functions outperformed ReLU on every transfer task. Normalization had little impact on the performance of $\mathrm{OR}_{*\mathrm{IL}}$, but a large impact on $\mathrm{XNOR}_{*\mathrm{IL}}$. The best overall performance was attained by the exact $\mathrm{XNOR}_{\mathrm{NIL}}$, closely followed by our approximate $\mathrm{XNOR}_{\mathrm{NAIL}}$ and ensembles containing either of these. Surprisingly, our approximation $\mathrm{OR}_{*\mathrm{AIL}}$ outperformed the exact form $\mathrm{OR}_{*\mathrm{IL}}$. The proposed activation functions were beaten only by PReLU, on Stanford Cars. On Caltech101, all MLPs were beaten by a linear layer, suggesting our MLPs were overfitting for that dataset. For further discussion, see Appendix § A.17.

4.6 Additional results

For results on tabular data, abstract reasoning and compositional zero-shot learning tasks, please see Appendix § A.12, Appendix § A.18 and Appendix § A.19, respectively.

5 Conclusion

In this work, we motivated and introduced novel activation functions analogous to Boolean operators in logit-space. We designed the AIL functions, fast approximates to the true logit-space probabilistic Boolean operations, and demonstrated their effectiveness on a wide range of tasks.

Although our activation functions assume independence (which is generally approximately true for pre-activation features learnt with 1D activations), we found the network learnt to induce anti-correlations between features which were paired together by our activations (Appendix § A.14). This suggests exact independence of the features is not essential to the performance of our proposed activations.

We found $\mathrm{XNOR}_{*\mathrm{AIL}}$ was highly effective for shallow MLPs. Meanwhile, $\mathrm{OR}_{\mathrm{AIL}}$ was highly effective for representation learning in the setting of a deep ResNet architecture trained on images. In scenarios which involve manipulating high-level features extracted by an embedding network, we find that using $\mathrm{XNOR}_{*\mathrm{IL}}$ or an ensemble of AIL activation functions together works best, and that the duplication ensembling strategy outperforms partitioning. In this work we restricted ourselves to

Table 1: Transfer learning from a frozen ResNet-18 architecture pretrained on ImageNet-1k to other computer vision datasets. Mean (std. error) of $n = 5$ random initializations of the MLP (same pretrained encoder). Bold: best. Underlined: top two. Italic: no sig. diff. from best (two-sided Student's t-test, $p > 0.05$). Background: linear color scale from ReLU baseline (white) to best (black).

	Test Accuracy (%)						
Activation function	Cal101	CIFAR10	CIFAR100	Flowers	StfCars	STL-10	SVHN
Linear layer only	***88.35***±0.15	78.56±0.09	57.39±0.09	92.32±0.20	33.51±0.06	94.68±0.02	46.60±0.14
ReLU	86.58±0.17	81.63±0.05	58.04±0.11	90.71±0.26	30.97±0.26	94.62±0.06	53.26±0.08
PReLU	*87.83*±0.21	81.03±0.13	58.90±0.18	*93.17*±0.19	***39.84***±0.18	94.54±0.05	53.47±0.08
SELU	87.74±0.09	79.93±0.13	58.24±0.06	92.27±0.13	37.51±0.17	94.53±0.07	50.94±0.12
GELU	87.10±0.15	81.39±0.09	58.51±0.13	91.51±0.15	33.43±0.15	94.62±0.06	53.43±0.23
Mish	87.11±0.12	81.09±0.11	58.37±0.10	91.61±0.15	33.75±0.14	94.61±0.05	53.05±0.12
Tanh	87.48±0.06	80.56±0.07	57.35±0.08	90.32±0.20	29.51±0.12	*94.63*±0.07	51.86±0.05
Max	86.96±0.20	81.76±0.14	58.60±0.12	90.98±0.18	33.37±0.15	*94.70*±0.06	53.53±0.16
Max, Min (d)	87.23±0.13	82.31±0.10	59.05±0.10	91.68±0.18	34.91±0.12	94.64±0.04	53.91±0.13
SignedGeomean	87.03±0.23	*51.45*±16.92	11.80±10.80	91.34±0.34	*26.37*±6.46	94.68±0.06	37.16±7.18
XNOR$_{\mathrm{IL}}$	85.01±0.17	79.62±0.09	57.14±0.07	84.76±0.43	1.34±0.11	94.51±0.03	51.99±0.16
OR$_{\mathrm{IL}}$	87.11±0.08	79.75±0.05	58.07±0.11	91.12±0.36	33.12±0.12	94.60±0.03	51.21±0.17
XNOR$_{\mathrm{NIL}}$	87.25±0.22	***82.88***±0.08	***60.78***±0.08	***93.26***±0.26	*39.47*±0.20	***94.83***±0.06	***55.34***±0.19
OR$_{\mathrm{NIL}}$	87.19±0.16	79.61±0.05	58.44±0.10	91.65±0.29	35.82±0.04	94.58±0.03	50.95±0.15
OR, AND$_{\mathrm{NIL}}$ (d)	86.82±0.18	80.09±0.09	58.60±0.07	91.44±0.20	37.03±0.11	94.65±0.05	52.49±0.09
OR, XNOR$_{\mathrm{NIL}}$ (d)	*87.82*±0.19	*82.67*±0.05	*60.60*±0.11	*92.93*±0.12	39.22±0.17	94.63±0.06	*54.87*±0.09
OR, AND, XNOR$_{\mathrm{NIL}}$ (d)	87.41±0.27	*82.84*±0.06	60.38±0.10	*92.98*±0.17	*39.42*±0.18	*94.71*±0.03	*55.11*±0.12
XNOR$_{\mathrm{AIL}}$	86.97±0.18	81.83±0.06	58.46±0.10	90.93±0.15	32.56±0.10	*94.71*±0.06	53.75±0.14
OR$_{\mathrm{AIL}}$	87.45±0.14	81.88±0.07	59.10±0.09	92.00±0.15	36.01±0.12	*94.69*±0.04	53.68±0.14
XNOR$_{\mathrm{NAIL}}$	87.61±0.23	82.38±0.07	59.77±0.13	*93.07*±0.20	*39.77*±0.04	*94.81*±0.03	53.91±0.05
OR$_{\mathrm{NAIL}}$	87.19±0.16	81.79±0.09	59.40±0.09	92.12±0.12	37.32±0.17	94.65±0.04	53.82±0.21
OR, AND$_{\mathrm{NAIL}}$ (d)	87.62±0.11	82.28±0.10	59.71±0.05	92.10±0.20	37.70±0.12	*94.61*±0.08	53.86±0.10
OR, XNOR$_{\mathrm{NAIL}}$ (d)	*87.85*±0.22	82.52±0.11	60.02±0.10	*93.12*±0.13	*39.64*±0.09	*94.75*±0.03	54.13±0.05
OR, AND, XNOR$_{\mathrm{NAIL}}$ (d)	87.78±0.14	*82.67*±0.06	60.01±0.21	*93.12*±0.21	*39.65*±0.14	*94.78*±0.03	54.58±0.12

only considering using a single activation function (or ensemble) throughout the network, however our results together indicate stronger results may be found by using OR$_{*\mathrm{AIL}}$ for feature extraction and either XNOR$_{*\mathrm{IL}}$ or ensemble $\{$OR, XNOR$_{*\mathrm{IL}}$ (d)$\}$ for later higher-order reasoning layers.

Our work shows there is more to learn about the importance of more complex activation functions, both for ANN applications and for non-linear dendritic integration in biological neuronal networks.

Acknowledgments and Disclosure of Funding

We are grateful to Chandramouli Shama Sastry, Eleni Triantafillou, and Finlay Maguire for insightful discussions.

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute `https://vectorinstitute.ai/partners/`, and in part by ACENET `https://ace-net.ca/` and Compute Canada `https://www.computecanada.ca/`. Additionally, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In L. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005.

Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 291–301. PMLR, 09–15 Jun 2019. doi:10.48550/arxiv.1811.05381.

David G. T. Barrett, Felix Hill, Adam Santoro, Ari S. Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 511–520. PMLR, 10–15 Jul 2018. doi:10.48550/arxiv.1807.04225.

Jonathan T. Barron. Continuously differentiable exponential linear units. *arXiv preprint*, arXiv:1704.07483, 2017. doi:10.48550/1704.07483.

Yaniv Benny, Niv Pekar, and Lior Wolf. Scale-localized abstract reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12557–12565, June 2021. doi:10.48550/arxiv.2009.09405.

Jock Blackard and Denis Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. volume 24, pp. 131–151, 12 1999. doi:10.1016/S0168-1699(99)00046-0.

Jock A. Blackard. *Comparison of Neural Networks and Discriminant Analysis in Predicting Forest Cover Types*. PhD thesis, Colorado State University, USA, 1998. AAI9921979.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 1881–1888, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Artem Chernodub and Dimitri Nowicki. Norm-preserving orthogonal permutation linear unit activation functions (OPLU). *arXiv preprint*, arXiv:1604.02313, 2017. doi:10.48550/arxiv.1604.02313.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint*, arXiv:1511.07289, 2016. doi:10.48550/arXiv.1511.07289.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation policies from data. *arXiv preprint*, arXiv:1805.09501, 2018. doi:10.48550/arXiv.1805.09501.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. ISSN 0893-6080. doi:10.1016/j.neunet.2017.12.012. Special issue on deep reinforcement learning.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006. doi:10.1109/TPAMI.2006.79.

François Fleuret, Ting Li, Charles Dubout, Emma K. Wampler, Steven Yantis, and Donald Geman. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences*, 108(43):17621–17625, 2011. ISSN 0027-8424. doi:10.1073/pnas.1109168108.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980. ISSN 1432-0770. doi:10.1007/BF00344251.

Albert Gidon, Timothy Adam Zolnik, Pawel Fidzinski, Felix Bolduan, Athanasia Papoutsi, Panayiota Poirazi, Martin Holtkamp, Imre Vida, and Matthew Evan Larkum. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87, 2020. doi:10.1126/science.aax6239.

Luke B. Godfrey and Michael S. Gashler. A parameterized activation function for learning fuzzy logic operations in deep neural networks. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 740–745, 2017. doi:10.1109/SMC.2017.8122696.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1319–1327, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: a steerable model for Bach chorales generation. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1362–1371. PMLR, 06–11 Aug 2017. doi:10.48550/arXiv.1612.01010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, New York, NY, USA, June 2016a. IEEE. doi:10.1109/CVPR.2016.90.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 630–645, Cham, 2016b. Springer International Publishing. ISBN 978-3-319-46493-0. doi:10.48550/arXiv.1603.05027.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint*, arXiv:1606.08415, 2020. doi:10.48550/1606.08415.

H. G. E. Hentschel, Alan Fine, and C. S. Pencea. Biological computing with diffusion and excitable calcium stores. *Mathematical Biosciences & Engineering*, 1(1):147–159, 06 2004. doi:10.3934/mbe.2004.1.147.

Hermann Hild, Johannes Feulner, and Wolfram Menzel. HARMONET: A neural net for harmonizing chorales in the style of J.S.Bach. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pp. 267–274, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558602224.

Sheng Hu, Yuqing Ma, Xianglong Liu, Yanlu Wei, and Shihao Bai. Stratified rule-aware network for abstract visual reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 1567–1574, May 2021. doi:10.48550/arxiv.2002.06838.

Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. *arXiv preprint*, arXiv:1903.07227, 2019. doi:10.48550/arXiv.1903.07227.

Phillip Isola, Joseph J Lim, and Edward H Adelson. Discovering states and transformations in image collections. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1383–1391, 2015. doi:10.1109/CVPR.2015.7298744.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, 2009. doi:10.1109/ICCV.2009.5459469.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1988–1997, July 2017. doi:10.1109/CVPR.2017.215.

H M Dipu Kabir, Moloud Abdar, Abbas Khosravi, Seyed Mohammad Jafar Jalali, Amir F Atiya, Saeid Nahavandi, and Dipti Srinivasan. SpinalNet: Deep neural network with gradual input. *IEEE Transactions on Artificial Intelligence*, pp. 1–13, 2022. doi:10.1109/TAI.2022.3185179.

Daniel Kahneman. *Thinking, fast and slow.* Thinking, fast and slow. Farrar, Straus and Giroux, New York, NY, US, 2011. ISBN 0-374-27563-7 (Hardcover); 1-4299-6935-0 (PDF); 978-0-374-27563-1 (Hardcover); 978-1-4299-6935-2 (PDF).

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint*, arXiv:1506.02078, 2015. doi:10.48550/arXiv.1506.02078.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. doi:10.48550/arxiv.1412.6980.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. doi:10.48550/1706.02515.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013. doi:10.1109/ICCVW.2013.77.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

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

Feynman Liang. Bachbot: Automatic composition in the style of Bach chorales. *University of Cambridge*, 8:19–48, 2016.

Michael London and Michael Häusser. Dendritic computation. *Annual Review of Neuroscience*, 28 (1):503–532, 2005. doi:10.1146/annurev.neuro.28.061604.135703.

Marvin Minsky and Seymour Papert. *Perceptrons.* Perceptrons. M.I.T. Press, Oxford, England, 1969.

Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint*, arXiv:1908.08681, 2019. doi:10.48550/arxiv.1908.08681.

Michael C Mozer. *Connectionist music composition based on melodic, stylistic, and psychophysical constraints*. University of Colorado, Boulder, Department of Computer Science, 1990.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve Restricted Boltzmann Machines. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 807–814. Omnipress, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, 2008. doi:10.1109/ICVGIP.2008.47.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi:10.23915/distill.00007.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi:10.23915/distill.00024.001.

D. B. Owen. A table of normal integrals. *Communications in Statistics - Simulation and Computation*, 9(4):389–419, 1980. doi:10.1080/03610918008812164.

Alexandre Payeur, Jean-Claude Béïque, and Richard Naud. Classes of dendritic information processing. *Current Opinion in Neurobiology*, 58:78–85, October 2019. ISSN 0959-4388. doi:https://doi.org/10.1016/j.conb.2019.07.006. Computational Neuroscience.

Senthil Purushwalkam, Maximilian Nickel, Abhinav Gupta, and Marc'Aurelio Ranzato. Task-driven modular networks for zero-shot compositional learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. doi:10.48550/arxiv.1905.05908.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint*, arXiv:1704.01444, 2017. doi:10.48550/arXiv.1704.01444.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint*, arXiv:1710.05941, 2017. doi:10.48550/1710.05941.

John C Raven and JH Court. *Raven's progressive matrices*. Western Psychological Services Los Angeles, CA, 1938.

Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of Naive Bayes text classifiers. In Tom Fawcett and Nina Mishra (eds.), *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 616–623. AAAI Press, 2003.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. *arXiv preprint*, arXiv:1803.09820, 2018. doi:10.48550/arxiv.1803.09820.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *arXiv preprint*, arXiv:1708.07120, 2017. doi:10.48550/arxiv.1708.07120.

UCI Machine Learning Repository. Covertype data set, Aug 1998. URL `https://archive.ics.uci.edu/ml/datasets/Covertype`.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. RAVEN: A dataset for relational and analogical visual reasoning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5312–5322, 2019. doi:10.1109/CVPR.2019.00546.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] The methodology we discuss is general-purpose and any societal impacts we could discuss would be highly general and completely speculative at this stage.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See §4.1–4.5 and Appendix § A.11–A.19.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All results figures and tables have error over several seeds, and the number of seeds used, clearly indicated.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix § A.9. The time to run each experiment is also indicated in the README files which accompany the code.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix § A.8, Appendix Table 3.

(b) Did you mention the license of the assets? [No] It is surprisingly difficult to discover the license under which popular ML datasets, such as CIFAR-10, are released! Most of the popular datasets within the community do not appear to be released under open licenses.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code to replicate our results is included in the supplemental material zip.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We describe the nature of the datasets we are using. There is no reason to expect that the data we are using contains personally identifiable information.

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]