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

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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

A Appendix

A.1 Approximating logistic AND and OR using ReLU

In Fig. 7 (upper panels), we show a representation of a 2D linear layer followed by the ReLU activation function. Changing projection used in the linear layer allows us to rotate and stretch the output only; each unit is, by construction, empty in half the plane.

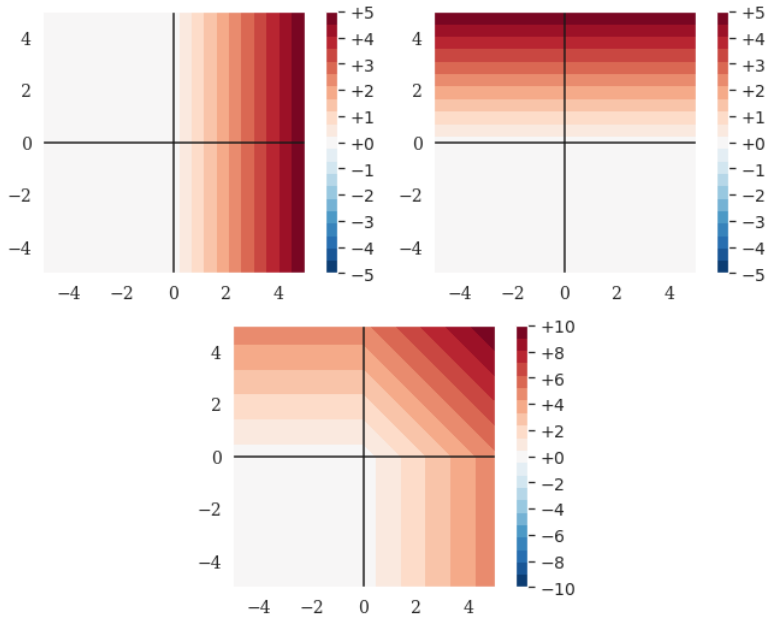


Figure 7: ReLU unit, and ReLU units followed by a linear layer to try to approximate $\mathrm{OR}_{\mathrm{IL}}$, leaving a dead space where the negative logits should be.

If we apply a second linear layer on top of the outputs of the first two units, we can try to approximate the logit AND or OR function with $z = \mathrm{ReLU}(x) + \mathrm{ReLU}(y)$. However, as shown in Fig. 7, lower panel, the solution using ReLU leaves a quadrant of the output space set to zero due to its behaviour at truncating away information. This illustrates why a ReLU-based neural network can not accurately approximate the logistic AND and OR functions between a pair of logits with only two hidden units. Comparing this with Figures 1 and 2 demonstrates the advantage of $\mathrm{AND}_{\mathrm{AIL}}$ and $\mathrm{OR}_{\mathrm{AIL}}$ in situations where the network needs to use probabilistic Boolean logic as its basis.

A.2 Solving XOR

A long-standing criticism of artificial neural networks is their inability to solve XOR with a single layer (Minsky & Papert, 1969). Of course, adding a single hidden layer allows a network using ReLU to solve XOR. However, the way that it solves the problem is to join two of the disconnected regions together in a stripe (see Fig. 8), which does not extrapolate to correctly solve the task beyond the bounds of the training data. Meanwhile, our $\mathrm{XNOR}_{\mathrm{NIL}}$ and $\mathrm{XNOR}_{\mathrm{NAIL}}$ activations are trivially able to solve the XOR problem without any hidden layers. For comparison here, we include one hidden layer with 2 units for each network. Including a layer before the activation function makes the task harder for networks using $\mathrm{XNOR}_{*\mathrm{IL}}$, activations which must learn how to project the input space in order to compute the desired separation. Also, including the linear layer allows the network to generalize to rotations and offset versions of the task.

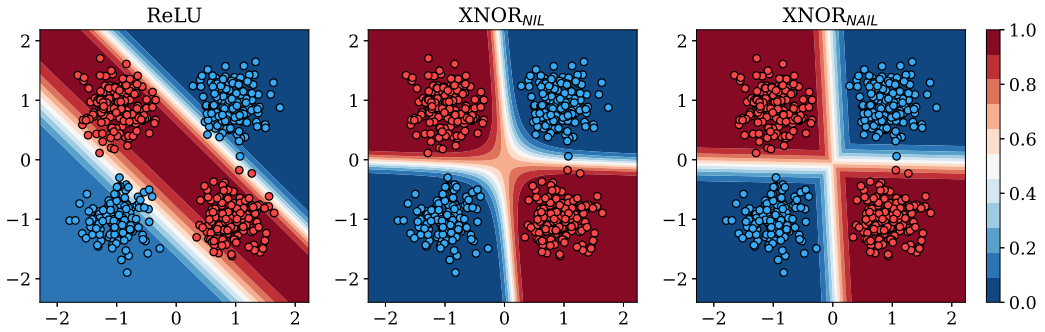


Figure 8: Solving XOR with a single hidden layer of 2 units, using either ReLU, $\mathrm{XNOR}_{\mathrm{NIL}}$, or $\mathrm{XNOR}_{\mathrm{NAIL}}$ activation. Circles indicate negative (blue) and positive (red) training samples. The heatmaps indicate the output probabilities of the networks.

A.3 Linear layers and Bayes' Rule in log-odds form

Bayes' Theorem or Bayes' Rule is given by

$$\mathrm{P}(H|X) = \frac{\mathrm{P}(X|H)\ \mathrm{P}(H)}{\mathrm{P}(X)}. \tag{8}$$

In this case, we update the probability of our hypothesis, H, based on the event of the observation of a new piece of evidence, X. Our prior belief for the hypothesis is $\mathrm{P}(H)$, and posterior is $\mathrm{P}(H|X)$. To update our belief from the prior and yield the posterior, we multiply by the Bayes factor for the evidence which is given by $\mathrm{P}(X|H)/\mathrm{P}(X)$.

Converting the probabilities into log-odds ratios (logits) yields the following representation of Bayes' Rule.

$$\log\left(\frac{\mathrm{P}(H|X)}{\mathrm{P}(H^C|X)}\right) = \log\left(\frac{\mathrm{P}(H)}{\mathrm{P}(H^C)}\right) + \log\left(\frac{\mathrm{P}(X|H)}{\mathrm{P}(X|H^C)}\right) \tag{9}$$

Here, H^C is the complement to H (the event that the hypothesis is false), and $\mathrm{P}(H^C) = 1 - \mathrm{P}(H)$. Our prior log-odds ratio is $\log\left(\mathrm{P}(H)/\mathrm{P}(H^C)\right)$, and our posterior after updating based on the observation of new evidence X is $\log\left(\mathrm{P}(H|X)/\mathrm{P}(H^C|X)\right)$. To update our belief from the prior and yield the posterior, we add the log-odds Bayes factor for the evidence which is given by $\log\left(\mathrm{P}(X|H)/\mathrm{P}(X|H^C)\right)$.

In log-odds space, a series of updates with multiple pieces of independent evidence can be performed at once with a summation operation.

$$\log\left(\frac{\mathrm{P}(H|\boldsymbol{x})}{\mathrm{P}(H^C|\boldsymbol{x})}\right) = \log\left(\frac{\mathrm{P}(H)}{\mathrm{P}(H^C)}\right) + \sum_i \log\left(\frac{\mathrm{P}(X_i|H)}{\mathrm{P}(X_i|H^C)}\right). \tag{10}$$

This operation can be represented by the linear layer in an artificial neural network, $z_k = b_k + \boldsymbol{w}_k^T \boldsymbol{x}$. Here, the bias term $b_k = \log\left(\mathrm{P}(H)/\mathrm{P}(H^C)\right)$ is the prior for hypothesis (the presence of the feature represented by the k-th neuron), and the series of weighted inputs from the previous layer, $w_{ki}\, x_i$ provide evidential updates. This is also equivalent to the operation of a multinomial naïve Bayes classifier, expressed in log-space, if we choose $w_{ki} = \log p_{ki}$ (Rennie et al., 2003).

A.4 Difference between AIL and IL functions

Here, we measure and show the difference between the true logit-space operations and our AIL approximations, shown in Fig. 9, Fig. 10, and Fig. 11.

In each case, we observe that the magnitude of the difference is never more than 1, which occurs along the boundary lines in AIL. Since the magnitude of the three functions increase as we move away from the origin, the relative difference decreases in magnitude as the size of x and y increase.

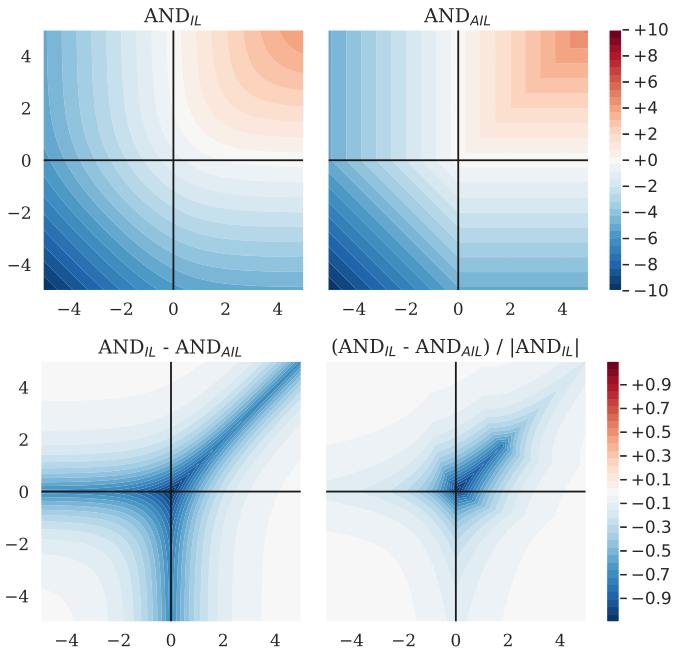


Figure 9: Heatmaps showing $\mathrm{AND}_{\mathrm{IL}}$, $\mathrm{AND}_{\mathrm{AIL}}$, their difference, and their relative difference.

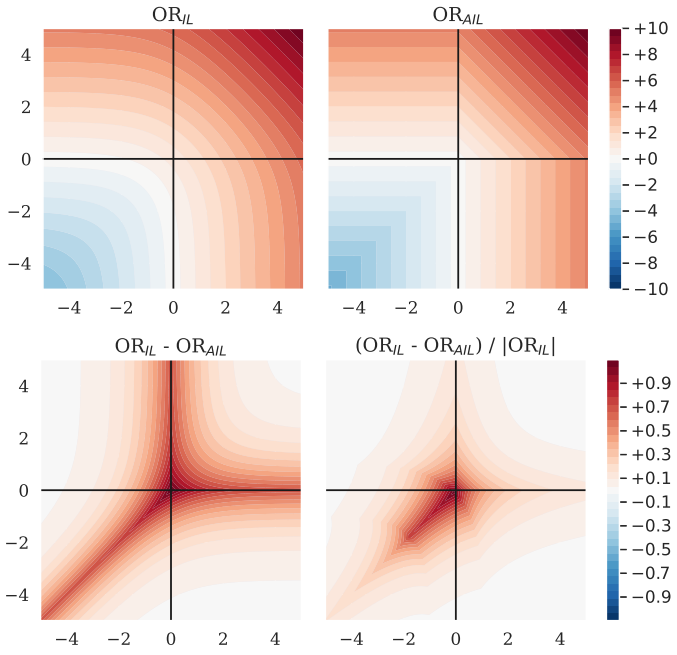


Figure 10: Heatmaps showing $\mathrm{OR}_{\mathrm{IL}}$, $\mathrm{OR}_{\mathrm{AIL}}$, their difference, and their relative difference.

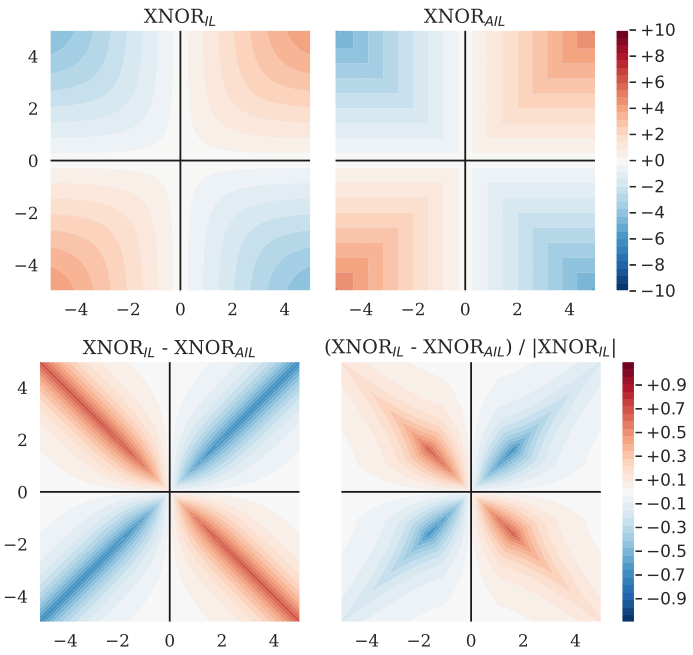


Figure 11: Heatmaps showing $\mathrm{XNOR}_{\mathrm{IL}}$, $\mathrm{XNOR}_{\mathrm{AIL}}$, their difference, and their relative difference.

A.5 Gradient of AIL and IL functions

We show the gradient of each of the logit-space Boolean operators and their AIL approximates in Fig. 12, Fig. 13, and Fig. 14. By the symmetry of each of the functions, the derivative with respect to y is a reflected copy of the gradient with respect to x.

We find that the gradient of each AIL function closely matches that of the exact form. Whilst there are "dead" regions where the gradient is zero, this only occurs for one of the derivatives at a time (there is always a gradient with respect to at least one of x and y).

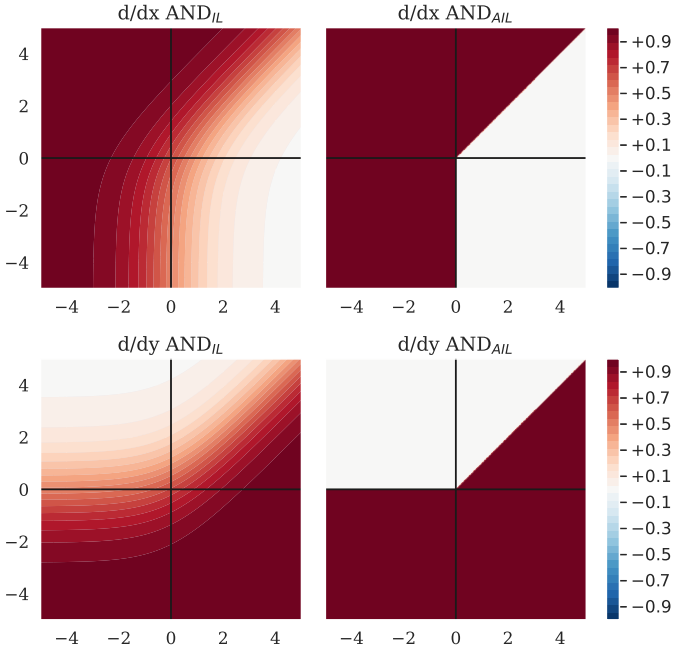


Figure 12: Heatmaps showing the gradient with respect to x and y of $\mathrm{AND}_{\mathrm{IL}}$ and $\mathrm{AND}_{\mathrm{AIL}}$.

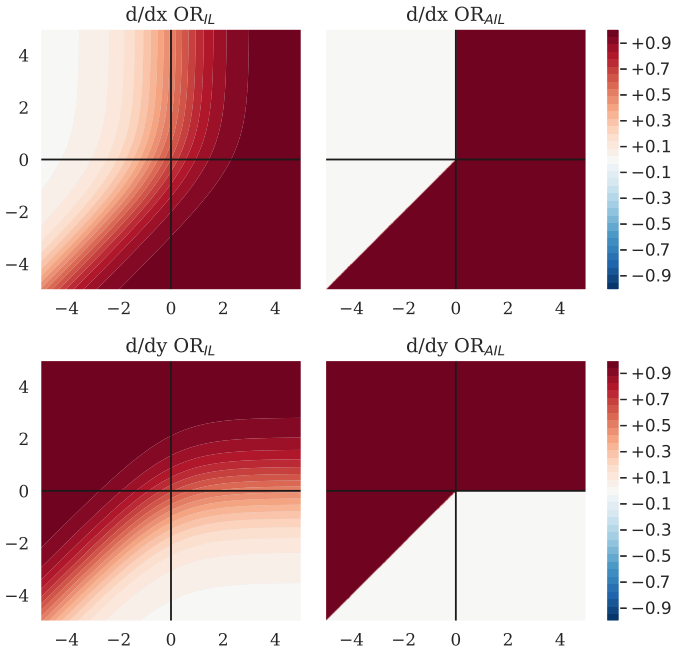


Figure 13: Heatmaps showing the gradient with respect to x and y of $\mathrm{OR}_{\mathrm{IL}}$ or $\mathrm{OR}_{\mathrm{AIL}}$.

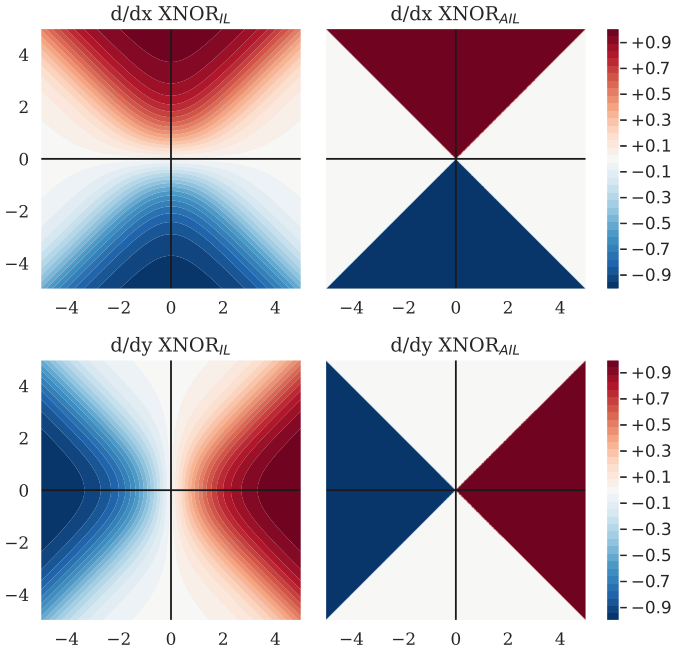
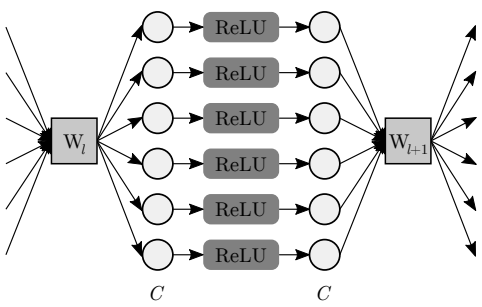


Figure 14: Heatmaps showing the gradient with respect to x and y of $\mathrm{XNOR}_{\mathrm{IL}}$ xnor $\mathrm{XNOR}_{\mathrm{AIL}}$.

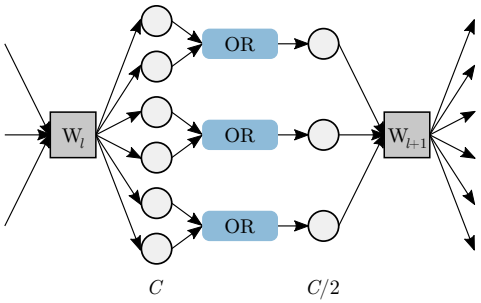
A.6 Activation pathing diagrams

Here we illustrate the implications of using a $2 \to 1$ activation function on the network achitecture, and the layout of the partition and duplication ensembling methods described in §3.5.

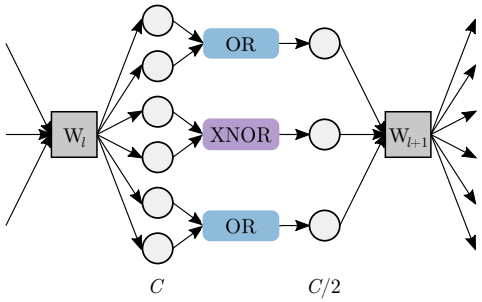
A standard 1D activation function with a $1 \to 1$ mapping such as ReLU processes each pre-activation logit independently, and the number of post-activation channels equals the pre-activation channels (Fig. 15a). The total number of parameters per layer is $C^2 + C \sim C^2$.



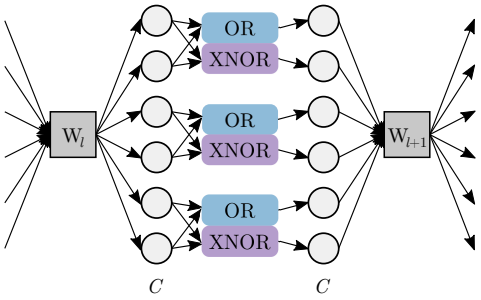
(a) ReLU layer ($1 \to 1$ activation function).



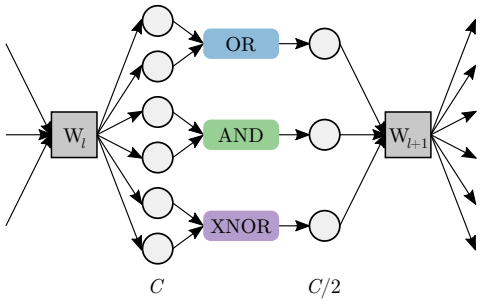
(b) $\mathrm{OR}_{\mathrm{AIL}}$ layer ($2 \to 1$ activation function).



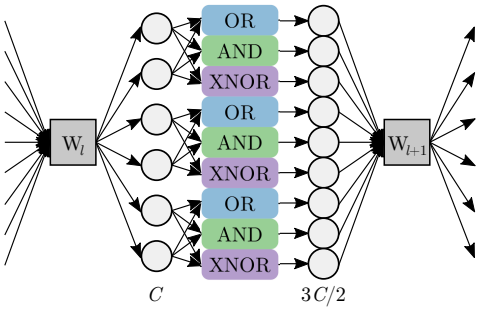
(c) Partition ensembling strategy, using OR and XNOR (overall $2 \to 1$).



(d) Duplication ensembling strategy, OR and XNOR (overall $2 \to 2$).



(e) Partition ensembling strategy, using OR, AND, and XNOR (overall $2 \to 1$).



(f) Duplication ensembling strategy, OR, AND and XNOR (overall $2 \to 3$).

Figure 15: Network architectures for one layer within an MLP network using various activation functions/ensembles (centered around the activation function). The preceding weight matrix for this layer, W_l, and the weight matrix for the next layer, W_{l+1}, each produce C features (channels). The activation functions may change the number of features, depending on their mapping.

For a $2 \to 1$ activation function such as $\mathrm{OR}_{\mathrm{AIL}}$, each activation function requires two arguments and produces a single output (Fig. 15b). The number of output channels is half that of the input, and there are $\frac{C^2}{2} + C \sim \frac{C^2}{2}$ parameters per layer.

If we use a combination of $2 \to 1$ activation functions with the partition ensembling strategy, each pre-activation neuron is used once, and only seen by one activation function. The number of output channels is again half that of the input, as seen in Fig. 15c and e.

Using the duplication ensembling strategy, each $2 \to 1$ activation function is applied in parallel to the same set of operands (Fig. 15d and f). The number of output channels scales up with the number of activation functions used in the ensemble. With two activation functions, the ensemble performs a $2 \to 2$ mapping; the number of channels is unchanged by the layer and the number of parameters per layer is equal to that of a $1 \to 1$ activation function, $C^2 + C \sim C^2$.

A.7 Normalization methodology

To normalize the IL and AIL functions, we assumed the pair of arguments to the activation function were drawn independently from the standard normal distribution, $\mathcal{N}(0,1)$, with probability density function

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right). \tag{11}$$

We found the expected value of the operator, and the standard deviation, and then subtracted by the mean and divided by the standard deviation,

$$\mathrm{OP}_{N*IL} = \frac{\mathrm{OP}_{*IL} - \mu}{\sigma}. \tag{12}$$

For the normalized version of the exact operators ($\mathrm{OR}_{\mathrm{NIL}}$, etc.), the mean and standard deviation were estimated empirically using 600 million samples. For our AIL functions, we determined the mean and standard deviation analytically by integration. The values are shown in Table 2.

Table 2: Mean and standard deviation of IL and AIL activation functions, for inputs drawn independently from the standard normal distribution.

Activation function	Mean	Standard deviation
$\mathrm{OR}_{\mathrm{IL}}$	1.298 95	0.948 34
$\mathrm{AND}_{\mathrm{IL}}$	−1.298 95	0.948 34
$\mathrm{XNOR}_{\mathrm{IL}}$	0	0.366 41
$\mathrm{OR}_{\mathrm{AIL}}$	0.681 04	0.972 29
$\mathrm{AND}_{\mathrm{AIL}}$	−0.681 04	0.972 29
$\mathrm{XNOR}_{\mathrm{AIL}}$	0	0.602 81

A.7.1 Derivations for $\mathrm{OR}_{\mathrm{AIL}}$

Here, we analytically derive the mean and variance of $\mathrm{OR}_{\mathrm{AIL}}$. Recall

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

which can be rewritten as

$$\mathrm{OR}_{\mathrm{AIL}}(x,y) = \begin{cases} x+y & \text{if } x>0,\ y>0 \\ x & \text{if } y\le 0,\ x\ge y \\ y & \text{if } x\le 0,\ x<y \end{cases} \tag{13}$$

Proposition A.1. *The expected value of* $\mathrm{OR}_{\mathrm{AIL}}(x,y)$ *for independently sampled* $x, y \sim \mathcal{N}(0,1)$ *is*

$$\mathbb{E}[\mathrm{OR}_{\mathrm{AIL}}(x,y)] = \frac{1}{\sqrt{2\pi}} + \frac{1}{2\sqrt{\pi}}. \tag{14}$$

Proof. Using Eq. 13,

$$\begin{aligned} \mathbb{E}[\mathrm{OR}_{\mathrm{AIL}}(x,y)] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{OR}_{\mathrm{AIL}}(x,y)\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \iint_{x>0,\,y>0} (x+y)\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &\quad + \iint_{x\le 0,\,x\ge y} x\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y + \iint_{y\le 0,\,x<y} y\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \end{aligned}$$

Let us consider the first component, where $x > 0\, y > 0$.

$$\begin{aligned} A &:= \iint_{x>0,\, y>0} (x+y)\, \phi(x)\, \phi(y)\, \mathrm{d}x\, \mathrm{d}y \\ &= \int_{x=0}^{\infty} \mathrm{d}x\, \phi(x) \int_{y=0}^{\infty} (x+y)\, \phi(y)\, \mathrm{d}y \\ &= \int_{x=0}^{\infty} \mathrm{d}x\, \phi(x) \int_{y=0}^{\infty} (x\, \phi(y) + y\, \phi(y))\, \mathrm{d}y \end{aligned}$$

Note that the integral of the probability density function $\phi(x)$ is its cumulative distribution function,

$$\int \phi(u)\ \mathrm{d}u = \Phi(u) + C \tag{15}$$

which has the limits $\lim_{u\to-\infty} \Phi(u) = 0$ and $\lim_{u\to+\infty} \Phi(u) = 1$, and central value $\Phi(0) = 0.5$.

The probability density function $\phi(x)$ has limits $\lim_{u\to-\infty} \phi(u) = \lim_{u\to+\infty} \phi(u) = 0$, and central value $\Phi(0) = {}^{1}/_{\sqrt{2\pi}}$.

From the integrals of normal densities, equation 11 given in Owen (1980), we have

$$\int u\, \phi(u)\, \mathrm{d}u = -\phi(u) + C. \tag{16}$$

Thus,

$$\begin{aligned} A &= \int_{x=0}^{\infty} \mathrm{d}x\, \phi(x) [x\, \Phi(y) - \phi(y)]_{y=0}^{\infty} \\ &= \int_{x=0}^{\infty} \mathrm{d}x\, \phi(x) \left[\frac{x}{2} + \frac{1}{\sqrt{2\pi}}\right] \\ &= \left[-\frac{1}{2}\phi(x) + \frac{1}{\sqrt{2\pi}}\Phi(x)\right]_{x=0}^{\infty} \\ &= \frac{1}{2\sqrt{2\pi}} + \frac{1}{\sqrt{2\pi}} - \frac{1}{2\sqrt{2\pi}} \\ &= \frac{1}{\sqrt{2\pi}} \end{aligned}$$

Next let us consider the second component of the integral.

$$\begin{aligned} B &:= \iint_{x\le 0,\, x\ge y} x\, \phi(x)\, \phi(y)\, \mathrm{d}x\, \mathrm{d}y \\ &= \int_{y=-\infty}^{0} \phi(x) \int_{x=y}^{\infty} \phi(y)\, x\, \mathrm{d}x\, \mathrm{d}y \\ &= \int_{y=-\infty}^{0} \mathrm{d}y\, \phi(y) \int_{x=y}^{\infty} x\, \phi(x)\, \mathrm{d}x \\ &= \int_{y=-\infty}^{0} \mathrm{d}y\, \phi(y) \left[-\phi(x)\right]_{x=y}^{\infty} \\ &= \int_{y=-\infty}^{0} \phi(y)^2\, \mathrm{d}y \\ &= \left[\frac{1}{2\sqrt{\pi}}\Phi(x\sqrt{2})\right]_{y=-\infty}^{0} \\ &= \frac{1}{4\pi} \end{aligned}$$

By change of variables,

$$\begin{aligned}\iint_{y\le 0,\, x<y} y\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y &= \iint_{x\le 0,\, y<x} x\,\phi(y)\,\phi(x)\,\mathrm{d}y\,\mathrm{d}x \\ &= B\end{aligned}$$

Thus we conclude

$$\begin{aligned}\mathbb{E}[\mathrm{OR}_{\mathrm{AIL}}(x,y)] &= A + 2B \\ &= \frac{1}{\sqrt{2\pi}} + \frac{2}{4\pi} \\ &= \frac{1}{\sqrt{2\pi}} + \frac{1}{2\pi}\end{aligned}$$

□

Proposition A.2. *The variance of* $\mathrm{OR}_{\mathrm{AIL}}(x,y)$ *for independently sampled* $x, y \sim \mathcal{N}(0,1)$ *is*

$$\mathrm{Var}(\mathrm{OR}_{\mathrm{AIL}}(x,y)) = \frac{5}{4} - \frac{1}{\sqrt{2\pi}} - \frac{1}{4\pi} \tag{17}$$

Proof. Using Eq. 13,

$$\begin{aligned}\mathbb{E}[\mathrm{OR}_{\mathrm{AIL}}(x,y)^2] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{OR}_{\mathrm{AIL}}(x,y)^2 \phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \iint_{x>0,\, y>0} (x+y)^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &\quad + \iint_{x\le 0,\, x^2\ge y} x^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y + \iint_{y\le 0,\, x<y} y^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y\end{aligned}$$

Let us consider the first component, where $x > 0,\ y > 0$.

$$\begin{aligned}A &:= \iint_{x>0,\, y>0} (x+y)^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \iint_{x>0,\, y>0} (x^2 + 2xy + y^2)\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \int_{x=0}^{\infty}\int_{y=0}^{\infty} (x^2 + 2xy + y^2)\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \int_{x=0}^{\infty}\int_{y=0}^{\infty} (x^2 + 2xy + y^2)\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y\end{aligned}$$

From equation 12 of Owen (1980), we note the identity

$$\int u^2\,\phi(u)\,\mathrm{d}u = \Phi(u) - u\,\phi(u) + C. \tag{18}$$

Additionally, we note that

$$\begin{aligned}\lim_{u\to\infty} u\,\phi(u) &= \lim_{u\to\infty} \frac{u}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) \\ &= 0\end{aligned}$$

Using this in addition to Eq. 16,

$$\begin{aligned}
A &= \int_{x=0}^{\infty} \mathrm{d}x \int_{y=0}^{\infty} x^2\,\phi(x)\,\phi(y) + 2xy\,\phi(x)\,\phi(y) + y^2\,\phi(x)\,\phi(y)\,\mathrm{d}y \\
&= \int_{x=0}^{\infty} \mathrm{d}x \left[x^2\,\phi(x)\,\Phi(y) - 2x\,\phi(x)\,\phi(y) + \phi(x)(\Phi(y) - y\,\phi(y))\right]_{y=0}^{\infty} \\
&= \int_{x=0}^{\infty} \mathrm{d}x \left[(x^2\,\phi(x) + \phi(x)) - \left(\frac{x^2\,\phi(x)}{2} - \frac{2\,x\,\phi(x)}{\sqrt{2\pi}} + \frac{\phi(x)}{2}\right)\right] \\
&= \int_{x=0}^{\infty} \mathrm{d}x \left(\frac{\phi(x)}{2} + \frac{\sqrt{2}\,x\,\phi(x)}{\sqrt{\pi}} + \frac{x^2\,\phi(x)}{2}\right) \\
&= \left[\frac{\Phi(x)}{2} - \frac{\sqrt{2}}{\sqrt{\pi}}\,\phi(x) + \frac{1}{2}(\Phi(x) - x\,\phi(x))\right]_{x=0}^{\infty} \\
&= \left(\frac{1}{2} + \frac{1}{2}\right) - \left(\frac{1}{4} - \frac{\sqrt{2}}{\sqrt{\pi}}\,\frac{1}{\sqrt{2\pi}} + \frac{1}{4}\right) \\
&= \frac{1}{2} + \frac{1}{\pi}
\end{aligned}$$

Next we consider the second component,

$$\begin{aligned}
B &:= \iint_{x\le 0,\, x\ge y} x^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\
&= \int_{y=-\infty}^{0} \int_{x=y}^{\infty} \phi(x)\,\phi(y)\,x^2\,\mathrm{d}x\,\mathrm{d}y \\
&= \int_{y=-\infty}^{0} \mathrm{d}y\,\phi(y) \int_{x=y}^{\infty} x^2\,\phi(x)\,\mathrm{d}x \\
&= \int_{y=-\infty}^{0} \mathrm{d}y\,\phi(y)\left[\Phi(x) - x\,\phi(x)\right]_{x=y}^{\infty} \\
&= \int_{y=-\infty}^{0} \mathrm{d}y\,\phi(y)\left[1 - \Phi(y) + y\,\phi(y)\right] \\
&= \int_{y=-\infty}^{0} \mathrm{d}y\,\left[\phi(y) - \phi(y)\,\Phi(y) + y\,\phi(y)^2\right]
\end{aligned}$$

From equation n1 of Owen (1980),

$$\begin{aligned}
\int u\,\phi(u)^n\,\mathrm{d}u &= \frac{-\phi\left(\sqrt{n}u\right)}{n(2\pi)^{(n-1)/2}} \\
\implies \int u\,\phi(u)^2\,\mathrm{d}u &= \frac{-\phi\left(\sqrt{2}u\right)}{2\sqrt{2\pi}}.
\end{aligned}$$

Additionally from equation 1,010.4 of Owen (1980),

$$\begin{aligned}
\int_{-\infty}^{0} \phi(au)\,\Phi(bu)\,\mathrm{d}u &= \frac{1}{2\pi|a|}\left(\frac{\pi}{2} - \arctan\left(\frac{b}{|a|}\right)\right) \\
\implies \int_{-\infty}^{0} \phi(u)\,\Phi(u)\,\mathrm{d}u &= \frac{1}{2\pi}\left(\frac{\pi}{2} - \frac{\pi}{4}\right) = \frac{1}{8}.
\end{aligned}$$

Thus

$$\begin{aligned}
B =& \int_{y=-\infty}^{0} \mathrm{d}y \, \left[\phi(y) - \phi(y)\,\Phi(y) + y\,\phi(y)^2\right] \\
=& -\frac{1}{8} + \left[\Phi(y) - \frac{\phi\left(\sqrt{2}y\right)}{2\sqrt{2\pi}}\right]_{y=-\infty}^{0} \\
=& -\frac{1}{8} + \frac{1}{2} - \frac{1}{\sqrt{2\pi}}\frac{1}{2\sqrt{2\pi}} \\
=& \frac{3}{8} - \frac{1}{4\pi}
\end{aligned}$$

By change of variables,

$$\begin{aligned}
\iint_{y\le 0,\, x<y} y^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y =& \iint_{x\le 0,\, y<x} x^2\,\phi(y)\,\phi(x)\,\mathrm{d}y\,\mathrm{d}x \\
=& B
\end{aligned}$$

Thus we conclude

$$\begin{aligned}
\mathbb{E}[\mathrm{OR}_{\mathrm{AIL}}(x,y)^2] &= A + 2B \\
&= \frac{1}{2} + \frac{1}{\pi} + \frac{3}{4} - \frac{1}{2\pi} \\
&= \frac{5}{4} + \frac{1}{2\pi}
\end{aligned}$$

Recall that

$$\begin{aligned}
\mathrm{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2.
\end{aligned}$$

Hence

$$\begin{aligned}
\mathrm{Var}(\mathrm{OR}_{\mathrm{AIL}}(x,y)) &= \frac{5}{4} + \frac{1}{2\pi} - \left(\frac{1}{\sqrt{2\pi}} + \frac{1}{2\sqrt{\pi}}\right)^2 \\
&= \frac{5}{4} + \frac{1}{2\pi} - \left(\frac{1}{2\pi} + 2\frac{1}{\sqrt{2\pi}}\frac{1}{2\sqrt{\pi}} + \frac{1}{4\pi}\right) \\
&= \frac{5}{4} + \frac{1}{2\pi} - \left(\frac{3}{4\pi} + \frac{1}{\sqrt{2}\pi}\right) \\
&= \frac{5}{4} - \frac{1}{\sqrt{2\pi}} - \frac{1}{4\pi}
\end{aligned}$$

□

A.7.2 Derivations for $\mathrm{AND}_{\mathrm{AIL}}$

Follows from $\mathrm{AND}_{\mathrm{AIL}}(x,y) = -\,\mathrm{OR}_{\mathrm{AIL}}(-x,-y)$.

A.7.3 Derivations for $\mathrm{XNOR}_{\mathrm{AIL}}$

Here, we analytically derive the mean and variance of $\mathrm{XNOR}_{\mathrm{AIL}}$. Recall

$$\mathrm{XNOR}_{\mathrm{AIL}}(x,y) := \mathrm{sgn}(xy)\min(|x|,|y|), \tag{19}$$

Proposition A.3. *The expected value of* $\mathrm{XNOR}_{\mathrm{AIL}}(x,y)$ *for independently sampled* $x, y \sim \mathcal{N}(0,1)$ *is* 0*.*

Proof. The expected value is given by

$$\mathbb{E}[\mathrm{XNOR}_{\mathrm{AIL}}(x,y)] = \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} \mathrm{XNOR}_{\mathrm{AIL}}(x,y)\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y. \tag{20}$$

Note that $\phi(x)$ is an even function, i.e. $\phi(x) = \phi(-x)$. Moreover, $\mathrm{XNOR}_{\mathrm{AIL}}(x,y)$ is an odd function with $\mathrm{XNOR}_{\mathrm{AIL}}(-x,y) = -\mathrm{XNOR}_{\mathrm{AIL}}(x,y)$. Thus $\mathrm{XNOR}_{\mathrm{AIL}}(x,y)\,\phi(x)\,\phi(y)$ is an odd function. Therefore,

$$\mathbb{E}[\mathrm{XNOR}_{\mathrm{AIL}}(x,y)] = 0 \tag{21}$$

□

Proposition A.4. *The variance of* $\mathrm{XNOR}_{\mathrm{AIL}}(x,y)$ *for independently sampled* $x, y \sim \mathcal{N}(0,1)$ *is*

$$\mathrm{Var}(\mathrm{XNOR}_{\mathrm{AIL}}(x,y)) = 1 - \frac{2}{\pi} \tag{22}$$

Proof. Recall that

$$\begin{aligned}\mathrm{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2.\end{aligned}$$

Using Proposition A.3,

$$\begin{aligned}\mathrm{Var}(\mathrm{XNOR}_{\mathrm{AIL}}(x,y)) &= \mathbb{E}[\mathrm{XNOR}_{\mathrm{AIL}}(x,y)^2] - \mathbb{E}[\mathrm{XNOR}_{\mathrm{AIL}}(x,y)]^2 \\ &= \mathbb{E}[\mathrm{XNOR}_{\mathrm{AIL}}(x,y)^2] - 0 \\ &= \mathbb{E}[\mathrm{sgn}(xy)^2\,\min(|x|,|y|)^2] \\ &= \mathbb{E}[\min(|x|,|y|)^2] \\ &= \mathbb{E}[\min(x^2,y^2)]\end{aligned}$$

since $x \in \mathbb{R}$, $y \in \mathbb{R}$.

$$\begin{aligned}\mathbb{E}[\mathrm{XNOR}_{\mathrm{AIL}}(x,y)^2] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\mathrm{XNOR}_{\mathrm{AIL}}(x,y))^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \min(x^2,y^2)\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &=: 8\,A\end{aligned}$$

The integrand has three symmetries: $f(-x) = f(x)$, $f(-y) = f(y)$, and $f(x,y) = f(y,x)$. Using this, we can break the integral up into 8 slices, each of which have an equal area, A.

We next integrate over one of these slices to find A.

$$\begin{aligned}A &= \int_{y=0}^{\infty}\int_{x=y}^{\infty} y^2\,\phi(x)\,\phi(y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \int_{y=0}^{\infty} y^2\,\phi(y)\,\mathrm{d}y \int_{x=y}^{\infty} \phi(x)\,\mathrm{d}x \\ &= \int_{y=0}^{\infty} y^2\,\phi(y)\,\mathrm{d}y\,\big[\Phi(\infty) - \Phi(y)\big] \\ &= \int_{y=0}^{\infty} y^2\,\phi(y)\,\big(1 - \Phi(y)\big)\,\mathrm{d}y \\ &= \int_{y=0}^{\infty} y^2\,\phi(y)\,\mathrm{d}y - \int_{y=0}^{\infty} y^2\,\phi(y)\,\Phi(y)\,\mathrm{d}y\end{aligned}$$

Equation (1;012-1) from Owen (1980) provides us with the identity

$$\int_{u=0}^{\infty} u^2\,\phi(u)\,\Phi(u)\,\mathrm{d}u = \frac{\Phi(u)^2}{2} - \frac{\phi(u)^2}{2} - u\,\phi(u)\,\Phi(u).$$

Thus,

$$
\begin{aligned}
A &= \left[\big(\Phi(y) - y\,\phi(y)\big) - \left(\frac{\Phi(y)^2}{2} - \frac{\phi(y)^2}{2} - y\,\phi(y)\,\Phi(y) \right) \right]_{y=0}^{\infty} \\
&= \left[\Phi(y) - y\,\phi(y) - \frac{\Phi(y)^2}{2} + \frac{\phi(y)^2}{2} + y\,\phi(y)\,\Phi(y) \right]_{y=0}^{\infty} \\
&= \left(1 - 0 - \frac{1}{2} + 0 + 0\right) - \left(\frac{1}{2} - 0 - \frac{1}{8} + \frac{1}{4\pi} + 0\right) \\
&= \frac{1}{8} - \frac{1}{4\pi} \\
\implies 8A &= 1 - \frac{2}{\pi} \\
&\implies \mathrm{Var}(\mathrm{XNOR}_{\mathrm{AIL}}(x, y)) = 1 - \frac{2}{\pi}
\end{aligned}
$$

□

Corollary A.4.1. *The standard deviation of* $\mathrm{XNOR}_{\mathrm{AIL}}(x, y)$ *for independently sampled* $x, y \sim \mathcal{N}(0, 1)$ *is*

$$\sigma = \sqrt{1 - \frac{2}{\pi}}. \tag{23}$$

A.8 Dataset summary

The datasets used in this work are summarized in Table 3.

Table 3: Dataset summaries.

	№ Samples			
Dataset	Train	Test	Classes	Reference
Bach Chorales	229	77	2	Boulanger-Lewandowski et al. (2012)
Caltech101	6162	1695	101	Fei-Fei et al. (2006)
CIFAR-10	50 000	10 000	10	Krizhevsky (2009)
CIFAR-100	50 000	10 000	100	Krizhevsky (2009)
Covertype	464 810	116 202	7	Blackard (1998); Blackard & Dean (1999)
I-RAVEN	6000	2000	—	Hu et al. (2021)
MIT-States	30 338	12 995	245 obj, 115 attr	Isola et al. (2015)
MNIST	60 000	10 000	10	LeCun et al. (1998)
Oxford Flowers	6552	818	102	Nilsback & Zisserman (2008)
Stanford Cars	8144	8041	196	Krause et al. (2013)
STL-10	5000	8000	10	Coates et al. (2011)
SVHN	73 257	26 032	10	Netzer et al. (2011)

We used the same splits for Caltech101 as used in Kabir et al. (2022).

The Covertype dataset was chosen as a dataset that contains only simple features (and not pixels of an image) on which we could study a simple MLP architecture, and was selected based on its popularity on the UCI ML repository.

The Bach Chorales dataset was chosen because — in addition to being in continued use by ML researchers for decades — it presents an interesting opportunity to consider a task where logical activation functions are intuitively applicable, as it is a relatively small dataset that has also been approached with rule-based frameworks, e.g. the expert system by Ebcioglu (1988).

MNIST, CIFAR-10, CIFAR-100 are standard image datasets, commonly used. We used small MLP and CNN architectures for the experiments on MNIST so we could investigate the performance of the network for many configurations (varying the size of the network). We used ResNet-50, a

very common deep convolutional architecture within the computer vision field, on CIFAR-10/100 to evaluate the performance in the context of a deep network.

The datasets used for the transfer learning task are all common and popular natural image datasets, with some containing coarse-grained classification (CIFAR-10), others fine-grained (Stanford Cars), and with a varying dataset size (5000—75 000 training samples). We chose to do an experiment involving transfer learning because it is a common practical situation where one must train only a small network that handles high-level features, and is the sort of situation which involves manipulating high-level features, relying on the pretrained network to do the feature extraction.

We considered other domains where logical reasoning is involved as a component of the task, and isolated abstract reasoning and compositional zero-shot learning as suitable tasks.

For abstract reasoning, we wanted to use an IQ style test, and determined that I-RAVEN was a state-of-the-art dataset within this domain (having fixed some problems with the pre-existing RAVEN dataset). We determined that the SRAN architecture from the paper which introduced I-RAVEN was still state-of-the-art on this task, and so used this.

Another problem domain in which we thought it would be interesting to study our activation functions was compositional zero-shot learning (CZSL). This is because the task inherently involves combining an attribute with an object (i.e. the AND operation). For CZSL, we looked at SOTA methods on https://paperswithcode.com. The best performance was from SymNet, but this was only implemented in TensorFlow and our code was set up in PyTorch so we built our experiments on the back of the second-best instead, which is the TMN architecture. In the TMN paper, they used two datasets: MIT-States and UT-Zappos-1. In our preliminary experiments, we found that the network started overfitting on MIT-States after around 6 epochs, but on UT-Zappos-1 it was overfitting after the first or second epoch (one can not tell beyond the fact the val performance is best for the first epoch). In the context of zero-shot learning, an epoch uses every image once, but there are also only a finite number of tasks in the dataset. Because there are multiple samples for each noun/adjective pair, and each noun only appears with a handful of adjectives and vice versa, there is in a way fewer tasks in one epoch than there are images. Hence it is possible for a zero-shot learning model to overfit to the training tasks in less than one epoch (recall that the network includes a pretrained ResNet model for extracting features from the images). For simplicity, we dropped UT-Zappos-1 and focused on MIT-States.

A.9 Experiment configuration and hardware

Experiments were run using PyTorch (typically v1.10) on an internal cluster of NVIDIA RTX-6000 and Tesla T4 GPUs. Our experiments each used either 1 or 4 GPUs.

A.10 Statistical Methodology

Here we provide information on how we carried out some of the detailed comparisons between activation functions.

In our experiments on Bach chorales (§4.2), MNIST (§4.3), CIFAR-10/100 (§4.4), and Covertype (§A.12), we explored the performance of networks using various activation functions whilst changing the number of parameters in the network. This was achieved by scaling up the width of the networks, whilst keeping the depth (number of layers) fixed.

Since the different structure of 1-to-1 and 2-to-1 activations (e.g. ReLU vs Max and the activations introduced in this paper) imply a different organization of the network, to explore the same range of values for the parameter count, the width values spanned different ranges depending on the activation function. Note that in other cases, it made more sense to preserve a characteristic such as the size of the embedding space, in which case we discuss this explicitly (e.g. §A.19).

In the cases where we explore a range of values for the parameter count, we adopt the following methodology for our statistical tests for the "best" activation function. First we collapsed across the number of params dimension by selecting the best width value for each activation function. Typically this is the widest value, but sometimes some activation functions had their performance peak with fewer parameters (especially on Covertype, where we had disabled weight decay). Then we identified the best performing activation function as the one with the highest accuracy.

For example, in the case of MLP on MNIST (§4.3), the best activation function (over all num. params considered) was $\mathrm{XNOR}_{\mathrm{AIL}}$. We compared its performance to each of the other activation functions using a (non-paired) two-tailed Student's t-test, implemented with `scipy.stats.ttest_ind`, one test for each of the other activations. The total number of tests performed is $(\text{num\_actfuns} - 1)$. We did not use a Bonferroni correction for multiple comparisons. In the case of the MLP on MNIST, the p-value vs $\mathrm{SignedGeomean}$ was between 0.05 and 0.1 and hence did not meet our threshold for statistical significance. For all the other activations, the p-value was less than 0.01, more than exceeding our threshold for significance. Hence, we found that $\mathrm{XNOR}_{\mathrm{AIL}}$ was significantly better than all other activations besides $\mathrm{SignedGeomean}$.

Note that this methodology (collapsing across the number of parameters by selecting the best for each activation function) compares each of the activation functions at their best. This is in keeping with how results are frequently reported in the literature (i.e. in a table showing one value per comparator, after performing a hyperparam search), and makes it possible to do a simple statistical test, such as a t-test. However, such a simplification does not tell the whole story about the results. For example, Fig. 5 also shows that for the MLP, XNOR-shaped activation functions also lose performance sooner than other activation functions when the number of parameters is reduced. Meanwhile for the CNN, some activation functions which perform highly with a large number of parameters maintain high performance even as the number of parameters is reduced, while others do not. This seemed to be associated with those activation functions that included Max/OR operations as opposed to those that did not. The methodology we used to evaluate the performance was the same as described above for the MLP. Selecting the best width for each activation function, we found there are 5 functions/ensembles which have indistinguishably best performance: (OR, AND, $XNOR_{AIL}$ (p), OR, XNOR AIL (d/p), Max, and SiLU) performed best, whilst other activations had significantly lower performance than top-performing activation function, $p < 0.05$.

A.11 Parity Experiments

Our parity experiment shown in §4.1 was trained for 100 epochs using ADAM, one-cycle learning rate schedule, max LR 0.01, weight decay 1×10^{-4}.

Following on from the parity experiment described in the main text, we also introduced a second synthetic dataset with a labelling function that, while slightly more complex than the first, was still solvable by applying the logical XNOR operation to the network inputs. In this dataset increased our number of inputs to 8, and derived our labels by applying a set of nested $\mathrm{XNOR}_{\mathrm{IL}}$ operations:

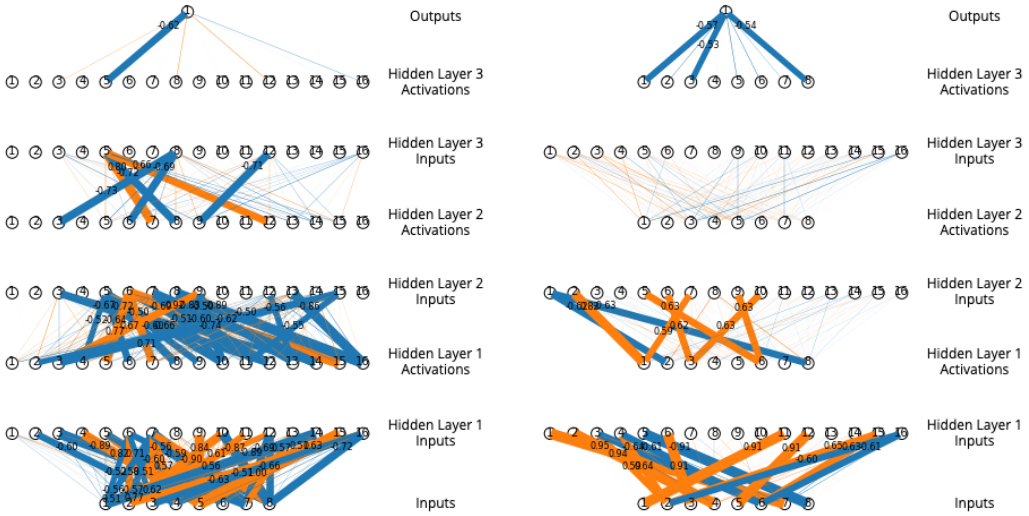
$$\begin{aligned}\mathrm{XNOR}_{\mathrm{IL}}(\,&\mathrm{XNOR}_{\mathrm{IL}}(\mathrm{XNOR}_{\mathrm{IL}}(x_2, x_5), \mathrm{XNOR}_{\mathrm{IL}}(x_3, x_4)),\\ &\mathrm{XNOR}_{\mathrm{IL}}(\mathrm{XNOR}_{\mathrm{IL}}(x_6, x_7), \mathrm{XNOR}_{\mathrm{IL}}(x_0, x_1)).\end{aligned}$$

For this more difficult task we also reformulated our initial experiment into a regression problem, as the continuous targets produced by this labelling function are more informative than the rounded binary targets used in the first experiment. We also adjusted our network setup to have an equal number of neurons at each hidden layer as we found that this significantly improved model performance[2]. We again trained using the same model hyperparameters for 100 epochs.

While this time the model was not able to learn a sparse weight matrix that exactly reflected our labelling function (see Fig. 16), the model was again able to leverage the $\mathrm{XNOR}_{\mathrm{AIL}}$ activation function to significantly outperform an identical model utilizing the ReLU activation function.

We found that a simple model with three hidden layers, each with eight neurons, utilizing $\mathrm{XNOR}_{\mathrm{AIL}}$ was able to go from a validation RMSE of 0.287 at the beginning of training to a validation RMSE of 0.016 after 100 epochs. Comparatively, an identical model utilizing the ReLU activation function was only able to achieve a validation RMSE of 0.271 after 100 epochs. In order for our ReLU network to match the validation RMSE of our 8-neuron-per-layer $\mathrm{XNOR}_{\mathrm{AIL}}$ model, we had to increase the model size by 32 times to 256 neurons at each hidden layer.

[2]We hypothesize that, because our $\mathrm{XNOR}_{\mathrm{AIL}}$ activation function reduces the number of hidden layer neurons by a factor of k, having a reduced number of neurons at each layer creates a bottleneck in the later layers of the network which restricts the amount of information that made its way through to the final layer.



(a) ReLU learned weight matrix

(b) XNOR_{AIL} learned weight matrix

Figure 16: Training results, regression experiment on second synthetic dataset.

A.12 MLP on Covertype

We trained small one, two, and three-layer MLP models on the Covertype dataset (UCI Machine Learning Repository, 1998; Blackard & Dean, 1999), a tabular dataset from the UCI Machine Learning Repository. The Covertype dataset is a classification task consisting of 581 012 samples of forest coverage across 7 classes. Each sample has 54 attributes.

Networks were trained using one-cycle (Smith & Topin, 2017; Smith, 2018) for 50 epochs, with a batch size of 1024, without weight decay. We used a fixed random 80:20 train:test split throughout our experiments. The learning rate was selected using an automated learning rate finder approach. No data augmentation was performed.

For each activation function, we varied the number of hidden units per layer to investigate how the activation functions affected the performance of the networks as its capacity changed. We did not use weight decay or data augmentation for this experiment, and so the network exhibits clear overfitting with larger architectures when the network is over-parameterized.

As shown in Fig. 17, we found that $\text{XNOR}_{\text{NAIL}}$ performed well on this task (outperforming ReLU), whilst OR_{NAIL} and Max were less successful, performing worse than ReLU. SignedGeomean performed well when using 1 hidden layer, but its relative performance drastically decreased as the MLP depth was increased. Tanh performed well throughout, and was best when using 2 hidden layers, but the margin vs $\text{XNOR}_{\text{NAIL}}$ was not statistically significant (two-sided Student's t-test, $p > 0.05$). SiLU performed poorly at this task, and the ELU family of activation functions (of which SELU is shown in the figures) performed even worse (reaching only $94.15 \pm 0.09\%$ with 3 hidden layers).

A.13 Bach Chorale training details

For each of the 4 voices, we restricted the available pitches to 3 octaves, resulting in 37 one-hot tokens (including silence). Since we only planned on feeding small time-windows of the chorale into our model, for pre-processing, we converted the data into a shape of $(L, 4, 37)$, where we set the sequence length to be $L = 4$.

To generate training examples for the discriminator, we transposed by $\{-5, -4, \ldots, 5, 6\}$ semitones, chosen uniformly at random, and there was a 0.5 probability that the sample was corrupted by the following method:

Table 4: MLP on Covertype, with 1–3 hidden layers, while varying the activation function and network width. For each activation, we show the best performance across all widths. Bold: best. Underlined: top two. Italic: no sig. diff. from best (two-sided Student's t-test, $p > 0.05$, $n = 5$ weight inits). Background: linear color scale from worst (white) to best (black) with a given number of layers.

		Test Accuracy (%) by № Layers		
Activation function	Map	1	2	3
ReLU	$1 \to 1$	$92.06_{\pm 0.07}$	$96.29_{\pm 0.02}$	$96.95_{\pm 0.01}$
LeakyReLU	$1 \to 1$	$92.12_{\pm 0.18}$	$96.24_{\pm 0.02}$	$96.93_{\pm 0.01}$
PReLU	$1 \to 1$	$92.78_{\pm 0.04}$	$96.67_{\pm 0.01}$	***97.12***$_{\pm 0.01}$
Softplus	$1 \to 1$	$86.46_{\pm 0.05}$	$93.51_{\pm 0.10}$	$95.50_{\pm 0.12}$
ELU	$1 \to 1$	$86.77_{\pm 0.14}$	$93.42_{\pm 0.05}$	$95.44_{\pm 0.06}$
CELU	$1 \to 1$	$86.71_{\pm 0.14}$	$93.42_{\pm 0.05}$	$95.44_{\pm 0.06}$
SELU	$1 \to 1$	$87.56_{\pm 0.04}$	$92.46_{\pm 0.10}$	$94.45_{\pm 0.12}$
GELU	$1 \to 1$	$90.74_{\pm 0.16}$	$96.09_{\pm 0.01}$	$96.91_{\pm 0.02}$
SiLU	$1 \to 1$	$88.90_{\pm 0.07}$	$95.47_{\pm 0.04}$	$96.73_{\pm 0.02}$
Hardswish	$1 \to 1$	$87.45_{\pm 0.05}$	$95.22_{\pm 0.05}$	$96.79_{\pm 0.03}$
Mish	$1 \to 1$	$89.24_{\pm 0.18}$	$95.47_{\pm 0.03}$	$96.77_{\pm 0.04}$
Softsign	$1 \to 1$	$94.24_{\pm 0.05}$	$96.61_{\pm 0.03}$	$96.85_{\pm 0.02}$
Tanh	$1 \to 1$	$\underline{94.59}_{\pm 0.03}$	***96.97***$_{\pm 0.02}$	*97.03*$_{\pm 0.04}$
GLU	$2 \to 1$	$93.97_{\pm 0.08}$	$96.85_{\pm 0.03}$	*97.09*$_{\pm 0.01}$
Max	$2 \to 1$	$91.67_{\pm 0.05}$	$95.99_{\pm 0.03}$	$96.95_{\pm 0.01}$
Max, Min (d)	$2 \to 2$	$92.06_{\pm 0.01}$	$96.06_{\pm 0.01}$	$96.94_{\pm 0.02}$
SignedGeomean	$2 \to 1$	***94.72***$_{\pm 0.02}$	$96.51_{\pm 0.06}$	$96.71_{\pm 0.02}$
XNOR_{IL}	$2 \to 1$	$92.23_{\pm 0.03}$	$96.64_{\pm 0.05}$	$97.04_{\pm 0.02}$
OR_{IL}	$2 \to 1$	$85.42_{\pm 0.06}$	$93.38_{\pm 0.05}$	$95.50_{\pm 0.06}$
OR, AND_{IL} (d)	$2 \to 2$	$84.79_{\pm 0.23}$	$92.83_{\pm 0.18}$	$95.19_{\pm 0.10}$
OR, XNOR_{IL} (p)	$2 \to 1$	$91.36_{\pm 0.08}$	$96.18_{\pm 0.05}$	$96.89_{\pm 0.01}$
OR, XNOR_{IL} (d)	$2 \to 2$	$89.64_{\pm 0.06}$	$95.34_{\pm 0.01}$	$96.51_{\pm 0.05}$
OR, AND, XNOR_{IL} (p)	$2 \to 1$	$90.45_{\pm 0.17}$	$95.60_{\pm 0.13}$	$96.66_{\pm 0.03}$
OR, AND, XNOR_{IL} (d)	$2 \to 3$	$88.59_{\pm 0.09}$	$94.51_{\pm 0.04}$	$96.05_{\pm 0.02}$
XNOR_{NIL}	$2 \to 1$	$90.53_{\pm 0.07}$	$96.23_{\pm 0.03}$	$96.93_{\pm 0.01}$
OR_{NIL}	$2 \to 1$	$85.08_{\pm 0.09}$	$93.09_{\pm 0.01}$	$95.32_{\pm 0.05}$
OR, AND_{NIL} (d)	$2 \to 2$	$85.16_{\pm 0.11}$	$92.86_{\pm 0.11}$	$95.35_{\pm 0.06}$
OR, XNOR_{NIL} (p)	$2 \to 1$	$89.99_{\pm 0.10}$	$95.82_{\pm 0.06}$	$96.67_{\pm 0.03}$
OR, XNOR_{NIL} (d)	$2 \to 2$	$88.63_{\pm 0.16}$	$95.37_{\pm 0.05}$	$96.62_{\pm 0.03}$
OR, AND, XNOR_{NIL} (p)	$2 \to 1$	$89.91_{\pm 0.18}$	$95.53_{\pm 0.06}$	$96.48_{\pm 0.06}$
OR, AND, XNOR_{NIL} (d)	$2 \to 3$	$87.99_{\pm 0.10}$	$94.76_{\pm 0.07}$	$96.16_{\pm 0.06}$
XNOR_{AIL}	$2 \to 1$	$94.22_{\pm 0.07}$	*96.90*$_{\pm 0.02}$	***97.12***$_{\pm 0.01}$
OR_{AIL}	$2 \to 1$	$91.14_{\pm 0.05}$	$95.74_{\pm 0.05}$	$96.83_{\pm 0.01}$
OR, AND_{AIL} (d)	$2 \to 2$	$91.32_{\pm 0.16}$	$95.85_{\pm 0.03}$	$96.87_{\pm 0.01}$
OR, XNOR_{AIL} (p)	$2 \to 1$	$93.66_{\pm 0.06}$	$96.60_{\pm 0.04}$	*97.11*$_{\pm 0.01}$
OR, XNOR_{AIL} (d)	$2 \to 2$	$93.58_{\pm 0.04}$	$96.60_{\pm 0.01}$	$97.06_{\pm 0.01}$
OR, AND, XNOR_{AIL} (p)	$2 \to 1$	$93.30_{\pm 0.05}$	$96.43_{\pm 0.03}$	*97.06*$_{\pm 0.02}$
OR, AND, XNOR_{AIL} (d)	$2 \to 3$	$93.09_{\pm 0.11}$	$96.32_{\pm 0.02}$	$96.99_{\pm 0.00}$
$\text{XNOR}_{\text{NAIL}}$	$2 \to 1$	$93.89_{\pm 0.03}$	$96.78_{\pm 0.01}$	$97.05_{\pm 0.01}$
OR_{NAIL}	$2 \to 1$	$91.57_{\pm 0.03}$	$95.93_{\pm 0.03}$	$96.84_{\pm 0.02}$
OR, AND_{NAIL} (d)	$2 \to 2$	$92.10_{\pm 0.09}$	$95.96_{\pm 0.05}$	$96.89_{\pm 0.03}$
OR, $\text{XNOR}_{\text{NAIL}}$ (p)	$2 \to 1$	$93.62_{\pm 0.05}$	$96.56_{\pm 0.02}$	$97.05_{\pm 0.01}$
OR, $\text{XNOR}_{\text{NAIL}}$ (d)	$2 \to 2$	$93.64_{\pm 0.07}$	$96.57_{\pm 0.01}$	$97.05_{\pm 0.01}$
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (p)	$2 \to 1$	$93.35_{\pm 0.06}$	$96.41_{\pm 0.01}$	$97.01_{\pm 0.01}$
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (d)	$2 \to 3$	$93.30_{\pm 0.02}$	$96.33_{\pm 0.02}$	$96.99_{\pm 0.01}$

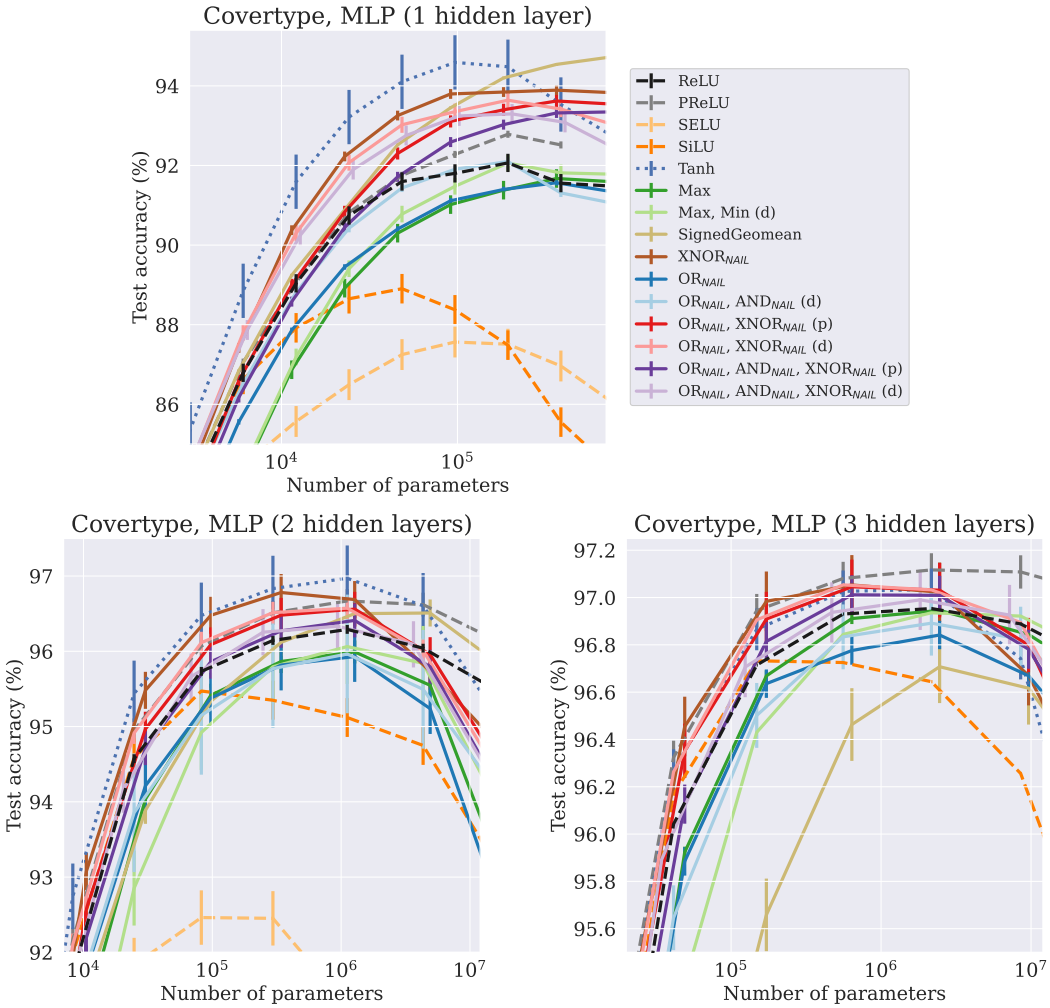


Figure 17: We trained MLPs on the Covertype dataset, with a fixed 80:20 random split. Trained with ADAM, 50 ep., 1-cycle, using LRs determined automatically with LR-finder. Mean (bars: std dev) of $n=5$ weight inits.

- Choose 2–3 notes in the $4\,L$ window to be corrupted
- For each note, corrupt by the following mixture distribution:
  - $(p = 0.6)$ Sample a pitch from a Gaussian centered on the existing note, with $\sigma = 3$ semitones, forcing the new pitch to be distinct
  - $(p = 0.2)$ Copy a pitch from the current voice, forcing the new pitch to be distinct
  - $(p = 0.2)$ Extend the previous note in time
  - $(p = 0.1)$ Sample uniformly from all 37 possible tokens

A.14 Correlations between pre-activations

To capture the existence of correlations, we took the cosine similarity between rows of the weight matrix. Since the inputs to all features in a given layer are the same, this is equivalent to measuring the similarity between corresponding pair of pre-activation features.

Results on correlations between weights in the JSB Chorale models are shown in Fig. 19. We found that when taking all pre-activations into account, every activation function generally showed independence between features. Interestingly, the cosine similarities between inputs that were paired

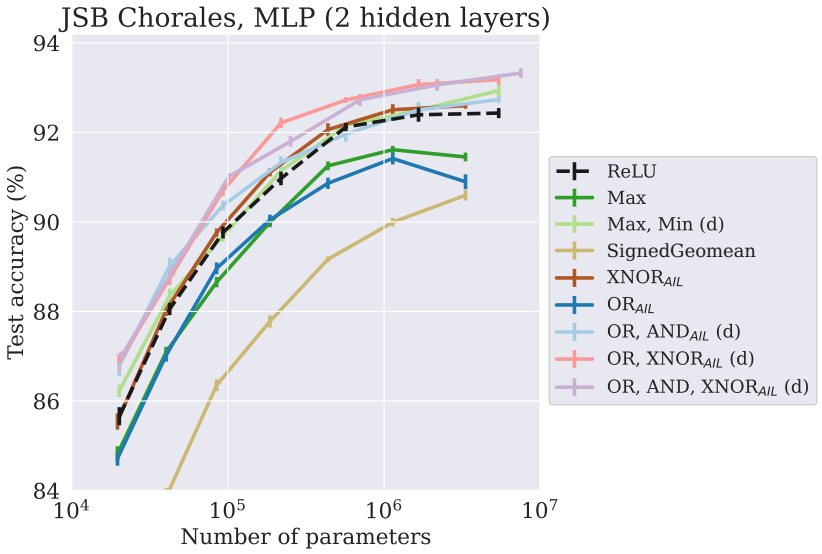


Figure 18: We trained 2-layer MLPs on JSB Chorales using ADAM (constant LR 1×10^{-3}, 150 ep.), Mean (bars: std dev) of $n = 10$ weight inits.

together for the bivariate activation functions showed anticorrelation in almost all cases where Max or $\mathrm{OR}_{\mathrm{AIL}}$ were used, and other cases generally showed more correlation than ReLU.

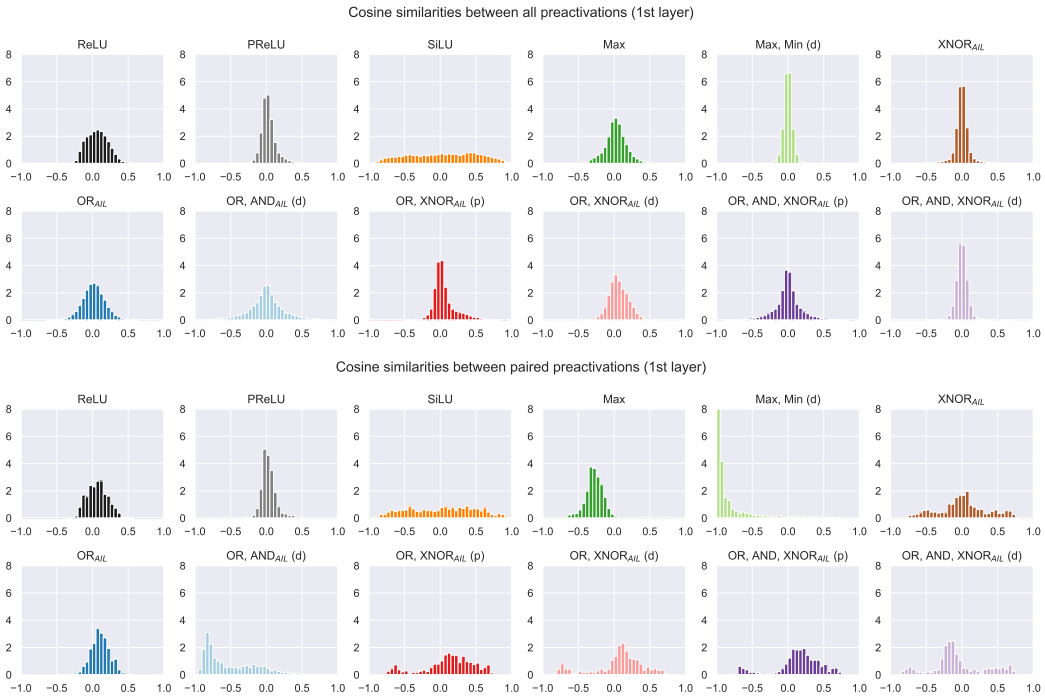


Figure 19: Cosine similarities between pre-activation weights of two activation functions in the first layer of an MLP trained on JSB Chorales.

We found that randomly selected pairs of preactivation features within the same layer have correlations that are given by a Gaussian-like distribution centered around zero. This was the case for all of the activation functions we tested. The behaviour of randomly selected pairs of features is thus reasonably consistent with the assumption of independence which we have made. We also investigated the

correlation between the pairs of preactivation features which were passing into our two-dimensional activation functions. Here, we found the correlation structure is different, and the correlation depends on the activation function being used. With Max and $\mathrm{OR}_{\mathrm{AIL}}$ activations, the network learns to make the columns of the weight matrix (and hence the preactivation scores for the pair of features) be inversely correlated. With $\mathrm{XNOR}_{\mathrm{AIL}}$, the network learns features which are either positively or negatively correlated (a wider distribution of correlations than seen with random pairs of features). We observe that (in all cases) the network learns to make the features passed to the AIL activation functions be correlated instead of independent, despite our assumption of independence. So it appears clear that the assumption of independence is violated, but also that it doesn't *really* matter because the network is choosing to break the assumption and induce these correlations between the features to get better performance.

A.15 CNN and MLP on MNIST

In this experiment we trained MLP and CNN models for 10 epochs on the MNIST dataset using ADAM optimizer, one-cycle learning rate schedule, and cross entropy loss, with batch size of 256. We augmented our training samples using a random affine transformation from: rotation of ± 10 degrees, scale of factor 0.8 to 1.2, translation with max absolute fraction for horizontal and vertical directions of 0.08, and shear parallel to the x-axes of ± 0.3.

The hyperparameters for the optimizer and scheduler were selected through a random search of the hyperparameter space. We chose to do random search instead of grid search because it typically yields better results for the same number of test cases.

For the hyperparameter search, we trained on the first $50\,000$ samples of the training partition and used the final $10\,000$ samples as a validation set. We ran the search for four iterations, with each iteration sampling 120 different hyperparameter settings. The initial bounds for our hyperparameter samples were set as described in Table 5. These bounds were chosen to be suitably wide such that the optimal configuration should be contained within them for all activation functions considered.

Table 5: Hyperparameter random search parameters.

Hyperparameter	Variable	Sampling (initial bounds)
ADAM beta1	$1 - 10^x$	$x \sim \mathrm{Uniform}(-3, -0.5)$
ADAM beta2	$1 - 10^x$	$x \sim \mathrm{Uniform}(-5, -1)$
ADAM epsilon	10^x	$x \sim \mathrm{Uniform}(-10, -6)$
Weight decay	10^x	$x \sim \mathrm{Uniform}(-7, -3)$
One-cycle max LR	10^x	$x \sim \mathrm{Uniform}(-4, 0)$
One-cycle peak	x	$x \sim \mathrm{Uniform}(0.1, 0.5)$

The bounds on our uniform random variable x were tightened at each iteration by selecting the top-5 performing hyperparameter settings, taking the mean $\bar{x}$ and weighted standard deviation σ across those settings, and re-setting the bounds for the next iteration to be equal to $\bar{x} \pm 1.5\,\sigma$. After the fourth iteration, we ran a final iteration where we selected the top-10 performing hyperparameter settings across the four previous iterations, and then re-ran these for another 120 seeds (i.e. random weight initializations). We then selected the hyperparameter settings which had the highest performance across all 120 seeds.

We found that the $\mathrm{XNOR}_{\mathrm{AIL}}$ activation function required quite different hyperparameters than the other activation functions, but $\mathrm{AND}_{\mathrm{AIL}}$ and Max activation functions used similar hyperparameters to ReLU. However, we have no reason to believe that the proposed AIL activation functions are more susceptible than others to the choice of hyperparameters or the way in which hyperparameters are selected.

Our MLP model consisted of two hidden layers of equal size with batch norm applied to each. The number of neurons in each layer was set, taking into consideration the current activation function being tested, to ensure the number of trainable parameters in the network remained static across experiments

Our CNN model consisted of six layers, where each layer was comprised of a set of 2D convolution filters (kernel size 3, stride 1, padding 1), batch normalization, and a non-linear activation. The network also applied a pooling layer (kernel size 2, stride 2) at the end of every second layer. After

the 6 CNN layers the network flattens the output and applies three linear layers. The number of output channels (i.e. pre-activations) for the six 2D convolutions were $[c, 2c, 4c, 4c, 8c, 8c]$, and the number of pre-activation neurons produced by the subsequent linear layers were $[32c, 16c]$. Similar to the MLP model, c was chosen to ensure the number of trainable parameters in the network remained fixed across experiments. When varying the number of network parameters in our experiments, if the number of parameters dropped below 1×10^6 in the CNN model, we changed the structure of the convolution layers and linear layers to $[c, c, c, 1.5c, 1.5c]$ and $[2c, 1.5c]$, respectively. This ensured that each layer had at least two channels for our activation functions to aggregate.

Once our optimal hyperparameters were selected for both our MLP and CNN models, we then trained each model several times with varying number of trainable parameters. The reason we vary the number of parameters in the network instead of the network size/structure is because a network using our logical activation functions can have a significantly different number of parameters than an identical network using ReLU activation, because our logical activations aggregate across neurons at each layer.

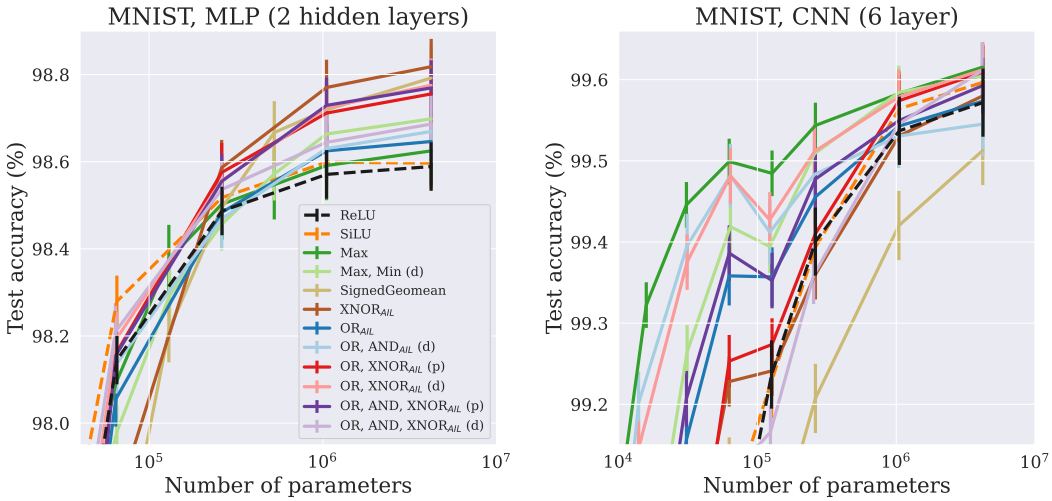


Figure 20: We trained CNN on MNIST, MLP on flattened-MNIST, using ADAM (1-cycle, 10 ep), hyperparams determined by random search. Mean (bars: std dev) of $n = 40$ weight inits.

Our abridged results are plotted in Fig. 5, with results for more activation functions shown in Fig. 20. Performance measurements from four slices of total parameter count are tabulated in Table 6 and Table 7.

A.16 ResNet50 on CIFAR-10/100

In this experiment, we explored the impact of the choice of activation functions on the performance of a pre-activation ResNet50 model (He et al., 2016a,b) applied to the CIFAR-10 and -100 datasets. Our base network was a pre-activation ResNet50 with 4 layers comprised of $[3, 4, 6, 3]$ bottleneck residual blocks (He et al., 2016b) with an expansion factor of 4. As described in the main text, we exchanged all ReLU activation functions in the network to a candidate activation function while maintaining the size of the pass-through embedding, with the exception of the activation function between the first convolutional layer (stem) and the first residual block which was ReLU throughout. We did not use any activation on the identity/pass-through branches of the residual blocks. The base widths for the residual blocks were $[64, 128, 256, 512]$, respectively. We experimented with changing the width of the network, scaling up the embedding space and all hidden layers by a common factor, w. We used width factors of 0.25, 0.5, 1, 2, and 4 for $1 \to 1$ activation functions (ReLU, etc), widths of 0.375, 0.75, 1.5, 3, and 6 for $2 \to 1$ activation functions (Max, etc), and widths of 0.2, 0.4, 0.75, 1.5, and 3 for $\{\text{OR}_{\text{AIL}}, \text{AND}_{\text{AIL}}, \text{XNOR}_{\text{AIL}}$ (d)$\}$. The stem width was held constant at 64 channels throughout.

For this experiment we trained a ResNet50 model for 100 epochs on CIFAR-10 and CIFAR-100 using ADAM optimizer, one-cycle learning rate schedule, cross entropy loss, and augmentations derived for

Table 6: MLP on MNIST, with two hidden layers, while varying the activation function and network width. 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 worst (white) to best (black) with a given number of parameters.

	Test Accuracy (%) by № Params.			
Activation function	∼16 k	∼65 k	∼262 k	∼4 M
ReLU	96.60±0.03	98.14±0.01	98.49±0.01	98.59±0.01
SiLU	***97.03***±0.02	***98.28***±0.01	98.52±0.01	98.60±0.01
Max	96.36±0.03	98.09±0.01	98.50±0.01	98.62±0.01
Max, Min (d)	96.38±0.03	97.98±0.02	98.46±0.01	98.70±0.01
SignedGeomean	91.03±0.15	97.59±0.02	98.49±0.01	*98.79*±0.01
XNOR_{AIL}	91.91±0.13	97.81±0.02	***98.59***±0.01	***98.82***±0.01
OR_{AIL}	96.22±0.03	98.05±0.01	98.48±0.01	98.65±0.01
OR, AND_{AIL} (d)	96.64±0.03	98.14±0.01	98.47±0.01	98.67±0.01
OR, XNOR_{AIL} (p)	95.85±0.03	98.16±0.01	*98.58*±0.01	98.76±0.01
OR, XNOR_{AIL} (d)	96.50±0.03	98.19±0.01	*98.56*±0.01	98.78±0.01
OR, AND, XNOR_{AIL} (p)	95.70±0.06	98.16±0.01	*98.55*±0.01	98.77±0.01
OR, AND, XNOR_{AIL} (d)	96.58±0.02	98.21±0.01	98.54±0.01	98.69±0.01

Table 7: CNN on MNIST, while varying the activation function and network width. 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 worst (white) to best (black) with a given number of parameters.

	Test Accuracy (%) by № Params.			
Activation function	∼15 k	∼63 k	∼260 k	∼4 M
ReLU	97.08±0.12	98.97±0.02	99.40±0.01	99.57±0.01
SiLU	97.08±0.14	99.07±0.01	99.39±0.01	99.60±0.01
Max	***99.32***±0.01	***99.50***±0.01	***99.54***±0.01	***99.62***±0.00
Max, Min (d)	98.90±0.02	99.42±0.01	99.51±0.01	*99.60*±0.01
SignedGeomean	98.28±0.03	99.12±0.01	99.21±0.01	99.51±0.01
XNOR_{AIL}	98.51±0.02	99.23±0.01	99.36±0.01	99.58±0.00
OR_{AIL}	98.83±0.02	99.36±0.01	99.46±0.01	99.57±0.01
OR, AND_{AIL} (d)	99.20±0.01	*99.48*±0.01	99.48±0.01	99.55±0.01
OR, XNOR_{AIL} (p)	98.34±0.04	99.25±0.01	99.41±0.01	*99.61*±0.01
OR, XNOR_{AIL} (d)	99.15±0.01	*99.48*±0.01	99.51±0.01	***99.61***±0.01
OR, AND, XNOR_{AIL} (p)	98.87±0.02	99.39±0.01	99.48±0.01	99.59±0.01
OR, AND, XNOR_{AIL} (d)	97.88±0.05	99.11±0.01	99.36±0.01	***99.61***±0.01

CIFAR-10 by AutoAugment (Cubuk et al., 2018), with batch size of 128. The hyperparameters for the optimizer and scheduler were determined through a random search of the hyper parameter space on the CIFAR-100 dataset for a fixed width factor $w = 2$ only. We used the same set of hyperparameters from this search for both the CIFAR-10 and CIFAR-100 experiments. The hyperparameter search was tuned against a partition of 10% of the CIFAR-100 training samples. Our hyperparameter search follows a similar approach to the one used for our MLP and CNN models on MNIST (§ A.15), but due to compute constraints with our larger models here we only trained 100 seeds each iteration, and only re-examined the top 10 seeds in the final round.

For PReLU, we did not perform a new hyperparameter search and simply re-used the same hyperparameters as discovered in the search with ReLU.

Our abridged results are plotted in Fig. 6, plotted in full with results for more activation functions shown in Fig. 21. Performance measurements from five slices of total parameter count are tabulated in Table 8 and Table 9.

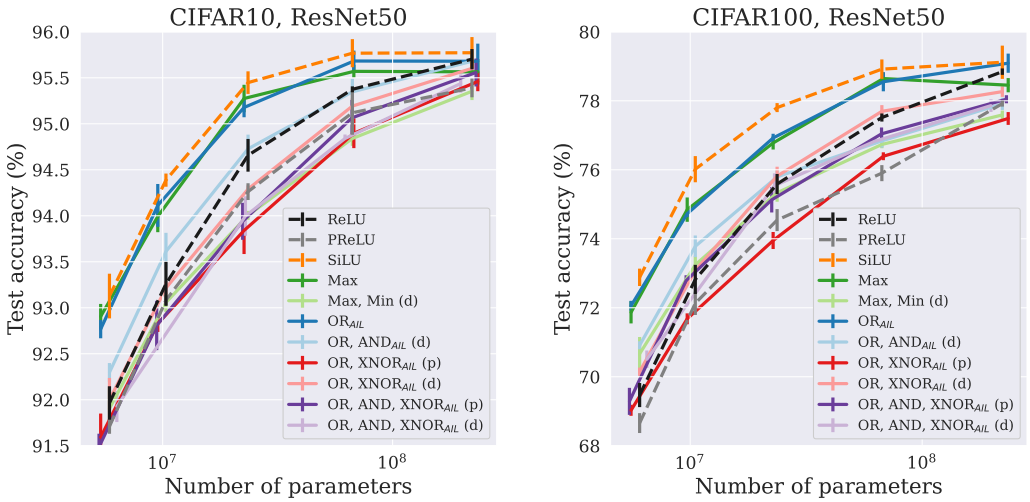


Figure 21: ResNet50 on CIFAR-10/100, varying the activation function used through the network. The width was varied to explore a range of network sizes (see text). Trained for 100 ep. with ADAM, using hyperparams as determined by random search on CIFAR-100 with width factor $w = 2$. Mean (bars: std dev) of $n = 4$ weight inits.

Table 8: ResNet50 on CIFAR-10, by number of parameters. Mean (std. error) of $n = 4$ random initializations. 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 second-lowest (white) to best (black) with a given number of parameters.

	Test Accuracy (%) by № Params.				
Activation function	∼6 M	∼10 M	∼23 M	∼67 M	∼225 M
ReLU	91.96±0.09	93.26±0.12	94.66±0.09	95.37±0.02	*95.70*±0.06
PReLU	91.71±0.04	93.08±0.03	94.26±0.05	95.12±0.12	95.39±0.05
SiLU	***93.13***±0.12	***94.39***±0.04	***95.45***±0.06	***95.77***±0.08	***95.77***±0.09
Max	*92.90*±0.07	93.99±0.09	*95.27*±0.07	*95.57*±0.04	*95.56*±0.07
Max, Min (d)	91.90±0.12	93.06±0.05	94.00±0.02	94.84±0.04	95.35±0.06
XNOR_{AIL}	88.97±0.11	88.93±0.20	90.02±0.29	90.77±0.53	89.96±0.34
OR_{AIL}	*92.77*±0.05	*94.11*±0.12	95.18±0.05	*95.68*±0.06	*95.68*±0.10
$\text{OR, AND}_{\text{AIL}}$ (d)	92.30±0.05	93.61±0.10	94.73±0.08	95.35±0.07	*95.68*±0.05
$\text{OR, XNOR}_{\text{AIL}}$ (p)	91.59±0.13	92.83±0.05	93.84±0.13	94.90±0.08	95.46±0.05
$\text{OR, XNOR}_{\text{AIL}}$ (d)	92.00±0.12	93.21±0.04	94.29±0.02	95.19±0.04	*95.60*±0.03
$\text{OR, AND, XNOR}_{\text{AIL}}$ (p)	91.49±0.07	92.81±0.13	93.94±0.10	95.06±0.10	*95.56*±0.08
$\text{OR, AND, XNOR}_{\text{AIL}}$ (d)	91.93±0.08	92.91±0.07	94.00±0.11	94.84±0.02	95.44±0.05

Table 9: ResNet50 on CIFAR-100, by number of parameters. Mean (std. error) of $n = 4$ random initializations. 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 second-lowest (white) to best (black) with a given number of parameters.

	Test Accuracy (%) by № Params.				
Activation function	$\sim$6 M	$\sim$10 M	$\sim$23 M	$\sim$67 M	$\sim$225 M
ReLU	69.47±0.17	72.82±0.21	75.59±0.15	77.51±0.06	*78.84*±0.05
PReLU	68.66±0.14	72.13±0.17	74.54±0.14	75.90±0.12	77.92±0.04
SiLU	***72.88***±0.13	***76.02***±0.19	***77.80***±0.07	***78.92***±0.14	***79.12***±0.24
Max	71.87±0.16	74.84±0.18	76.78±0.09	*78.64*±0.09	*78.45*±0.10
Max, Min (d)	70.66±0.24	73.24±0.12	75.33±0.12	76.73±0.05	77.59±0.06
XNOR_{AIL}	62.08±0.35	63.77±0.35	64.36±0.31	66.30±0.24	68.60±0.31
OR_{AIL}	72.06±0.07	74.72±0.05	76.92±0.06	*78.55*±0.12	*79.09*±0.12
OR, AND_{AIL} (d)	70.91±0.06	73.78±0.14	75.79±0.10	76.84±0.09	77.91±0.08
OR, XNOR_{AIL} (p)	69.03±0.08	71.68±0.08	73.95±0.12	76.39±0.06	77.49±0.09
OR, XNOR_{AIL} (d)	70.14±0.06	73.09±0.06	75.82±0.12	77.69±0.09	78.27±0.07
OR, AND, XNOR_{AIL} (p)	69.29±0.20	72.79±0.08	75.13±0.18	77.04±0.09	78.04±0.06
OR, AND, XNOR_{AIL} (d)	70.51±0.13	73.17±0.06	75.54±0.15	76.82±0.05	77.86±0.11

A.17 Transfer learning

We used a pretrained ResNet50 model taken from the PyTorch hub. The 2-layer MLP head was trained for 25 epochs on each dataset. Since we are interested in transfer learning in a data-limited regime for these experiments, we used only a small amount of data augmentation. We applied horizontal flip ($p=0.5$), scaling (0.7 to 1), aspect ratio stretching ($^3/_4$ to $^4/_3$), and colour jitter (intensity 0.4) only. For the SVHN dataset, horizontal flip was not performed. Pixel intensity normalization was done against the ImageNet mean and standard deviation.

The MLP head was optimized using SGD, momentum 0.9. We used a batch size of 128, maximum learning rate 0.01, and weight decay 1×10^{-4}. Before training began, we passed one epoch worth of inputs through the network without updating the weights in order to refresh the batch normalization statistics to the new dataset. We also performed one epoch of training with a warmup learning rate of 1×10^{-5} before commencing training at the maximum learning rate of 0.01. The learning rate was decayed with a cosine annealing schedule over 24 epochs. We report the performance of the final model at the end of the 25 epochs.

We used a pre-activation width of 512 neurons for ReLU and other and activation functions which map $1 \to 1$. To approximately match up the number of parameters, we used a width of 650 for activation functions which map $2 \to 1$, and 438 for $\{\mathrm{OR}, \mathrm{AND}, \mathrm{XNOR}_{\mathrm{AIL}}$ (d)$\}$ which maps features $3 \to 2$. These values control the total number of parameters in the head to be the same for datasets with 100 classes (the median number of classes in the datasets we considered). Our results with widths 438/512/650 and $\sim$600k parameters are shown in the main results, Table 1. To investigate a comprehensive set of baselines, we compared against every activation function implemented in PyTorch v1.10. This was too many comparisons to include in the main results, so for brevity we narrowed the activation functions down by omitting some activation functions where there were several similar functions and kept the better performing one (SiLU vs Mish, LeakyReLU vs PReLU, etc). The full set of results is shown in Table 10. Additionally, the precise widths and number of trainable parameters used for these experiments are shown Table 11.

Since the transfer learning experiments were performed without retraining the base network, we found (Table 1) the performance is limited by the features which the base network produces, whose relevance depends on the overlap between the domain of the pretraining task (ImageNet) and the new task. This information bottleneck, and variation in its utility, means the variation between performance on different datasets is much larger than the variation for different activation functions within the same dataset.

For transfer tasks which involve coarse-grained discrimination on images which are similar to ImageNet, the embedding generated by the pretrained model is already sufficient to separate the classes in the new dataset. Examples of this are Caltech101, where linear layer beats out using additional layers with non-linearities; and STL-10, where our best model is on-par with the linear model. The results on these two transfer learning tasks are too simple to be of interest to study. The performance is limited by the information retained by the pretrained embedding, but it appears that what task-relevant information is there is readily available with a linear layer without needing additional logic to interpret it.

Other transfer learning tasks we attempted are less trivial and show a larger difference in performance across the activation functions, and a gap from the linear layer readout model. The Stanford Cars dataset involves fine-grained discrimination between different car models, for which features generated by a model pretrained on ImageNet are not effective. The SVHN dataset, which contains images of house numbers, is coarse-grained but uses images which are outside the domain of ImageNet, which makes the transfer-learning task more difficult.

As shown in Table 10, we found the duplication ensemble strategy consistently outperformed the partition strategy on the transfer learning experiments. Compared to using $\mathrm{OR}_{\mathrm{NAIL}}$, there was no benefit to using an ensemble with the partition strategy. However, it was beneficial to ensemble $\mathrm{OR}_{\mathrm{NAIL}}$ with $\mathrm{AND}_{\mathrm{NAIL}}$, and better still with $\mathrm{XNOR}_{\mathrm{NAIL}}$, under the duplication strategy. The performance of duplication-ensembles using $\mathrm{XNOR}_{\mathrm{NAIL}}$, i.e. $\{\mathrm{OR}_{\mathrm{NAIL}}, \mathrm{AND}_{\mathrm{NAIL}}, \mathrm{XNOR}_{\mathrm{NAIL}}$ (d)$\}$ and $\{\mathrm{OR}_{\mathrm{NAIL}}, \mathrm{AND}_{\mathrm{NAIL}}, \mathrm{XNOR}_{\mathrm{NAIL}}$ (d)$\}$, was comparable to using $\mathrm{XNOR}_{\mathrm{NAIL}}$ alone, so in this case an ensemble was neither beneficial nor detrimental.

Additionally, we ran the experiment again with a pre-activation width of $w = 512$ for all activation functions. The results with constant pre-activation width $w = 512$ are shown in Table 12. By using

Table 10: 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
LeakyReLU	86.60±0.13	81.67±0.11	58.01±0.09	90.73±0.32	31.09±0.24	94.61±0.05	53.24±0.10
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
Softplus	86.16±0.18	79.13±0.08	56.58±0.07	89.39±0.29	21.23±0.13	94.63±0.03	48.79±0.08
ELU	87.18±0.09	80.44±0.08	58.08±0.10	91.71±0.14	34.70±0.06	94.55±0.05	51.89±0.09
CELU	87.18±0.09	80.44±0.08	58.08±0.10	91.71±0.14	34.70±0.06	94.55±0.05	51.89±0.09
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
SiLU	86.91±0.11	80.53±0.11	58.14±0.12	91.37±0.18	32.15±0.17	94.59±0.05	52.35±0.16
Hardswish	87.12±0.12	80.10±0.10	58.25±0.10	91.56±0.25	33.17±0.23	94.62±0.05	51.82±0.13
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
Softsign	81.47±0.18	80.03±0.09	54.84±0.09	82.34±0.22	17.33±0.10	94.70±0.03	51.25±0.08
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
GLU	86.71±0.31	79.19±0.07	57.64±0.10	90.34±0.19	27.04±0.12	94.57±0.03	50.12±0.19
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_{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_{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
OR, AND_{IL} (d)	87.22±0.14	79.56±0.06	57.91±0.12	91.44±0.18	36.04±0.07	94.44±0.05	50.42±0.25
OR, XNOR_{IL} (p)	86.31±0.24	80.29±0.10	58.13±0.08	90.51±0.20	31.41±0.11	94.76±0.02	52.78±0.13
OR, XNOR_{IL} (d)	86.76±0.12	80.83±0.08	58.55±0.10	90.76±0.07	33.56±0.14	94.47±0.07	52.54±0.09
OR, AND, XNOR_{IL} (p)	86.61±0.10	80.33±0.06	58.12±0.04	90.88±0.31	32.40±0.12	94.74±0.04	52.69±0.10
OR, AND, XNOR_{IL} (d)	86.78±0.10	80.18±0.02	58.70±0.17	91.68±0.07	34.82±0.14	94.49±0.04	52.09±0.18
XNOR_{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_{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_{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_{NIL} (p)	87.65±0.21	*82.67*±0.08	60.24±0.18	*92.52*±0.20	38.86±0.29	94.78±0.02	*55.25*±0.11
OR, XNOR_{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_{NIL} (p)	87.50±0.17	82.33±0.11	59.99±0.06	*92.62*±0.35	38.41±0.10	94.81±0.03	*54.91*±0.08
OR, AND, XNOR_{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_{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_{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
OR, AND_{AIL} (d)	87.43±0.11	82.38±0.06	59.90±0.08	92.07±0.18	37.16±0.15	94.55±0.05	54.05±0.07
OR, XNOR_{AIL} (p)	87.06±0.19	82.08±0.03	59.29±0.10	91.47±0.35	35.07±0.18	94.81±0.03	54.41±0.19
OR, XNOR_{AIL} (d)	87.09±0.21	82.20±0.04	59.44±0.07	91.90±0.10	36.88±0.10	94.69±0.06	54.03±0.11
OR, AND, XNOR_{AIL} (p)	87.22±0.12	82.10±0.03	59.34±0.13	91.71±0.19	35.65±0.11	94.77±0.05	54.32±0.12
OR, AND, XNOR_{AIL} (d)	87.49±0.11	82.50±0.08	59.83±0.12	92.37±0.08	37.60±0.20	94.72±0.02	54.55±0.09
$\text{XNOR}_{\text{NAIL}}$	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_{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_{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, $\text{XNOR}_{\text{NAIL}}$ (p)	87.61±0.15	82.37±0.11	59.95±0.13	*92.67*±0.10	38.60±0.08	*94.88*±0.03	54.49±0.13
OR, $\text{XNOR}_{\text{NAIL}}$ (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, $\text{XNOR}_{\text{NAIL}}$ (p)	87.48±0.14	82.27±0.08	59.87±0.05	92.10±0.29	38.26±0.12	***94.95***±0.02	54.42±0.13
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (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

a constant pre-activation width, the number of parameters used is not consistent across activation functions. The number of trainable parameters in each experiment is shown in Table 13. Note: some experiments on SVHN are missing due to an issue with the job scheduler which was not noticed in time to run the missing experiments.

We found that reducing the width from 650 to 512 (and hence reducing total number of trainable parameters) reduced the performance of $2\to1$ activation functions XNOR_{AIL}, OR_{AIL}, {OR, XNOR_{AIL} (p)}, and {OR, AND, XNOR_{AIL} (p)}. Increasing the width from 438 to 512 increased the performance of {OR, AND, XNOR_{AIL} (d)}.

Table 11: Hidden pre-activation widths and number of trainable parameters used in transfer learning experiments (corresponding to results shown in Table 1).

			№ Trainable Parameters						
Activation function	Map	Width	Cal101	CIFAR10	CIFAR100	Flowers	StfCars	STL-10	SVHN
Linear layer only			52k	5k	51k	52k	101k	5k	5k
ReLU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
LeakyReLU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
PReLU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
Softplus	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
ELU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
CELU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
SELU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
GELU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
SiLU	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
Hardswish	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
Mish	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
Softsign	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
Tanh	$1 \to 1$	512	577k	530k	577k	578k	626k	530k	530k
GLU	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
Max	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
Max, Min (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
SignedGeomean	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
$\mathrm{XNOR}_{\mathrm{IL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
$\mathrm{OR}_{\mathrm{IL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
OR, $\mathrm{AND}_{\mathrm{IL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, $\mathrm{XNOR}_{\mathrm{IL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, $\mathrm{XNOR}_{\mathrm{IL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, $\mathrm{XNOR}_{\mathrm{IL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, AND, $\mathrm{XNOR}_{\mathrm{IL}}$ (d)	$2 \to 3$	438	579k	519k	579k	580k	642k	519k	519k
$\mathrm{XNOR}_{\mathrm{NIL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
$\mathrm{OR}_{\mathrm{NIL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
OR, $\mathrm{AND}_{\mathrm{NIL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, $\mathrm{XNOR}_{\mathrm{NIL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, $\mathrm{XNOR}_{\mathrm{NIL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, $\mathrm{XNOR}_{\mathrm{NIL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, AND, $\mathrm{XNOR}_{\mathrm{NIL}}$ (d)	$2 \to 3$	438	579k	519k	579k	580k	642k	519k	519k
$\mathrm{XNOR}_{\mathrm{AIL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
$\mathrm{OR}_{\mathrm{AIL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
OR, $\mathrm{AND}_{\mathrm{AIL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, $\mathrm{XNOR}_{\mathrm{AIL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, $\mathrm{XNOR}_{\mathrm{AIL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, $\mathrm{XNOR}_{\mathrm{AIL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, AND, $\mathrm{XNOR}_{\mathrm{AIL}}$ (d)	$2 \to 3$	438	579k	519k	579k	580k	642k	519k	519k
$\mathrm{XNOR}_{\mathrm{NAIL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
$\mathrm{OR}_{\mathrm{NAIL}}$	$2 \to 1$	650	578k	549k	578k	579k	609k	549k	549k
OR, $\mathrm{AND}_{\mathrm{NAIL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, $\mathrm{XNOR}_{\mathrm{NAIL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, $\mathrm{XNOR}_{\mathrm{NAIL}}$ (d)	$2 \to 2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, $\mathrm{XNOR}_{\mathrm{NAIL}}$ (p)	$2 \to 1$	648	576k	546k	576k	576k	607k	546k	546k
OR, AND, $\mathrm{XNOR}_{\mathrm{NAIL}}$ (d)	$2 \to 3$	438	579k	519k	579k	580k	642k	519k	519k

Table 12: Transfer learning from a frozen ResNet-18 architecture pretrained on ImageNet-1k to other computer vision datasets. As with Table 1, but in this case we show results where the MLP has a pre-activation width of $w = 512$. Note that although the pre-activation width is constant, the number of parameters in the network is not consistent between experiments. The number of parameters is shown in Table 13. Mean (standard error) of $n = 5$ inits of the MLP (same pretrained network).

	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
LeakyReLU	86.60±0.13	81.67±0.11	58.01±0.09	90.73±0.32	31.09±0.24	94.61±0.05	53.24±0.10
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
Softplus	86.16±0.18	79.13±0.08	56.58±0.07	89.39±0.29	21.23±0.13	94.63±0.03	48.79±0.08
ELU	87.18±0.09	80.44±0.08	58.08±0.10	91.71±0.14	34.70±0.06	94.55±0.05	51.89±0.09
CELU	87.18±0.09	80.44±0.08	58.08±0.10	91.71±0.14	34.70±0.06	94.55±0.05	51.89±0.09
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
SiLU	86.91±0.11	80.53±0.11	58.14±0.12	91.37±0.18	32.15±0.17	94.59±0.05	52.35±0.16
Hardswish	87.12±0.12	80.10±0.10	58.25±0.10	91.56±0.25	33.17±0.23	94.62±0.05	51.82±0.13
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
Softsign	81.47±0.18	80.03±0.09	54.84±0.09	82.34±0.22	17.33±0.10	94.70±0.03	51.25±0.08
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
GLU	86.34±0.16	79.35±0.05	57.22±0.12	90.10±0.20	26.72±0.13	94.63±0.05	50.64±0.10
Max	86.86±0.11	81.56±0.06	58.12±0.10	90.59±0.25	32.80±0.04	94.65±0.05	53.55±0.09
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	86.74±0.33	23.79±13.79	11.75±10.75	90.98±0.21	32.45±0.15	94.59±0.04	19.59±0.00
XNOR_{IL}	85.11±0.10	79.77±0.08	56.92±0.09	84.29±0.29	1.78±0.21	94.56±0.05	52.27±0.08
OR_{IL}	87.01±0.21	79.88±0.04	57.68±0.08	91.12±0.10	32.55±0.04	94.57±0.06	51.54±0.07
OR, AND_{IL} (d)	87.22±0.14	79.56±0.06	57.91±0.12	91.44±0.18	36.04±0.07	94.44±0.05	50.42±0.25
OR, XNOR_{IL} (p)	86.74±0.21	80.41±0.06	57.85±0.08	90.39±0.25	30.98±0.17	94.63±0.04	52.77±0.04
OR, XNOR_{IL} (d)	86.76±0.12	80.83±0.08	58.55±0.10	90.76±0.07	33.56±0.14	94.47±0.07	52.54±0.09
OR, AND, XNOR_{IL} (p)	86.58±0.15	80.31±0.08	57.99±0.10	90.39±0.28	31.76±0.16	94.66±0.04	52.52±0.06
OR, AND, XNOR_{IL} (d)	86.98±0.24	80.22±0.10	58.89±0.10	91.56±0.26	35.27±0.08	94.49±0.05	52.53±0.04
XNOR_{NIL}	87.94±0.08	*82.89*±0.09	60.22±0.12	*93.11*±0.22	38.93±0.12	94.78±0.03	55.23±0.11
OR_{NIL}	87.23±0.14	79.76±0.05	58.62±0.02	91.47±0.07	35.57±0.11	94.66±0.07	51.18±0.05
OR, AND_{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_{NIL} (p)	87.60±0.16	82.61±0.05	59.90±0.16	92.32±0.29	38.31±0.14	94.69±0.03	55.01±0.06
OR, XNOR_{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_{NIL} (p)	87.23±0.14	82.36±0.09	59.42±0.12	92.32±0.02	37.92±0.20	94.77±0.04	54.60±0.08
OR, AND, XNOR_{NIL} (d)	87.33±0.26	***83.11***±0.10	***60.90***±0.16	*93.15*±0.28	*39.27*±0.18	*94.71*±0.09	***55.77***±0.13
XNOR_{AIL}	86.42±0.13	81.74±0.05	57.88±0.09	90.50±0.17	31.55±0.11	*94.78*±0.04	53.77±0.04
OR_{AIL}	87.28±0.09	81.81±0.02	58.68±0.05	91.78±0.23	35.28±0.09	94.67±0.07	53.68±0.03
OR, AND_{AIL} (d)	87.43±0.11	82.38±0.06	59.90±0.08	92.07±0.18	37.16±0.15	94.55±0.05	54.05±0.07
OR, XNOR_{AIL} (p)	87.24±0.22	81.94±0.03	58.57±0.07	90.93±0.17	34.59±0.18	*94.77*±0.06	53.89±0.11
OR, XNOR_{AIL} (d)	87.09±0.21	82.20±0.04	59.44±0.07	91.90±0.10	36.88±0.10	94.69±0.06	54.03±0.11
OR, AND, XNOR_{AIL} (p)	87.02±0.11	81.83±0.04	58.93±0.16	91.17±0.29	34.86±0.15	94.75±0.04	53.92±0.09
OR, AND, XNOR_{AIL} (d)	87.26±0.15	82.45±0.08	60.20±0.11	92.41±0.17	37.89±0.14	94.65±0.04	55.05±0.09
$\text{XNOR}_{\text{NAIL}}$	87.47±0.12	82.38±0.05	59.34±0.08	*92.84*±0.18	39.03±0.05	***94.88***±0.02	54.23±0.15
OR_{NAIL}	87.33±0.14	81.78±0.03	59.27±0.06	91.83±0.14	36.79±0.20	*94.78*±0.07	53.79±0.15
OR, AND_{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, $\text{XNOR}_{\text{NAIL}}$ (p)	87.79±0.10	82.26±0.05	59.27±0.10	92.18±0.19	38.20±0.15	*94.82*±0.05	54.09±0.09
OR, $\text{XNOR}_{\text{NAIL}}$ (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, $\text{XNOR}_{\text{NAIL}}$ (p)	87.34±0.26	82.11±0.18	59.56±0.10	92.27±0.22	37.57±0.10	*94.78*±0.04	54.12±0.08
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (d)	87.42±0.10	82.83±0.05	*60.58*±0.15	*92.54*±0.20	*39.49*±0.15	*94.81*±0.11	55.10±0.04

Table 13: Number of trainable parameters used in transfer learning experiments (corresponding to results shown in Table 12).

			№ Trainable Parameters						
Activation function	Map	Width	Cal101	CIFAR10	CIFAR100	Flowers	StfCars	STL-10	SVHN
Linear layer only			52k	5k	51k	52k	101k	5k	5k
ReLU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
LeakyReLU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
PReLU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
Softplus	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
ELU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
CELU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
SELU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
GELU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
SiLU	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
Hardswish	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
Mish	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
Softsign	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
Tanh	$1\to1$	512	577k	530k	577k	578k	626k	530k	530k
GLU	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
Max	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
Max, Min (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
SignedGeomean	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
XNOR_{IL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR_{IL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, AND_{IL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, XNOR_{IL} (p)	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, XNOR_{IL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, XNOR_{IL} (p)	$2\to1$	510	418k	395k	418k	418k	442k	395k	395k
OR, AND, XNOR_{IL} (d)	$2\to3$	512	734k	664k	733k	735k	807k	664k	664k
XNOR_{NIL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR_{NIL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, AND_{NIL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, XNOR_{NIL} (p)	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, XNOR_{NIL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, XNOR_{NIL} (p)	$2\to1$	510	418k	395k	418k	418k	442k	395k	395k
OR, AND, XNOR_{NIL} (d)	$2\to3$	512	734k	664k	733k	735k	807k	664k	664k
XNOR_{AIL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR_{AIL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, AND_{AIL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, XNOR_{AIL} (p)	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, XNOR_{AIL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, XNOR_{AIL} (p)	$2\to1$	510	418k	395k	418k	418k	442k	395k	395k
OR, AND, XNOR_{AIL} (d)	$2\to3$	512	734k	664k	733k	735k	807k	664k	664k
$\text{XNOR}_{\text{NAIL}}$	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR_{NAIL}	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, AND_{NAIL} (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, $\text{XNOR}_{\text{NAIL}}$ (p)	$2\to1$	512	420k	397k	420k	420k	445k	397k	397k
OR, $\text{XNOR}_{\text{NAIL}}$ (d)	$2\to2$	512	577k	530k	577k	578k	626k	530k	530k
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (p)	$2\to1$	510	418k	395k	418k	418k	442k	395k	395k
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (d)	$2\to3$	512	734k	664k	733k	735k	807k	664k	664k

A.18 Abstract reasoning

Abstract reasoning is challenging for neural networks to learn because their structure and objective function are more effective for tasks which come instinctively to humans ("System 1" of Kahneman, 2011), such as object recognition, as opposed to demanding ("System 2") logic tasks.

In recent years, several challenges have been proposed to evaluate the ability of neural networks to perform abstract reasoning. The Raven's Progressive Matrices (Raven & Court, 1938) are a long-standing IQ test, which have been emulated by Barrett et al. (2018) with Procedurally Generated Matrices (PGM), and then the RAVEN task by (Zhang et al., 2019). Recently Hu et al. (2021) and Benny et al. (2021) have improved on that task with I-RAVEN and RAVEN-FAIR, respectively, both aiming to make the task more balanced. Other abstract reasoning tasks have also been proposed (Fleuret et al., 2011; Johnson et al., 2017; Barrett et al., 2018).

We considered the application of AIL activation functions in the context of the I-RAVEN task, by adapting the Stratified Rule-Aware Network (SRAN) of Hu et al. (2021) to include our AIL activation functions. We first added LayerNorm to the network, and then swapped out the seven ReLU activations in the gating module. The architecture for the three ResNet-18 (He et al., 2016a) base models were unchanged. Where necessary, the number of units per layer was modified to facilitate the change in dimensionality caused by our activation function. The networks were trained using the same procedure as described by Hu et al. (2021).

Table 14: Performance of SRAN-based models on the I-RAVEN dataset (Hu et al., 2021). Bold: best. Underlined: second best. Background: color scale from worst in to best, linear with accuracy value.

		I-RAVEN Test Acc (%)							
Activation function	Params	Acc	Center	2×2G	3×3G	O-IC	O-IG	L-R	U-D
ReLU, Base SRAN (Hu et al., 2021)	44.0M	60.8	78.2	**50.1**	42.4	68.2	46.3	70.1	70.3
ReLU, SRAN+LayerNorm	45.6M	63.0	84.0	**50.0**	42.5	69.9	48.3	73.5	72.3
PReLU	45.6M	54.5	69.2	45.4	40.5	64.4	47.6	57.5	57.3
CELU	45.6M	56.6	73.5	46.5	41.0	66.6	46.1	62.8	59.7
SELU	45.6M	53.5	67.2	43.8	40.1	62.8	44.5	58.0	58.1
GELU	45.6M	61.4	80.8	49.2	42.5	69.2	48.3	70.3	69.2
SiLU	45.6M	59.4	78.0	47.2	41.4	66.3	47.8	69.1	66.2
Max	44.6M	57.8	76.3	45.2	39.7	65.3	48.7	64.6	64.7
Max, Min (d)	45.6M	60.2	80.2	49.5	41.9	66.5	47.0	68.5	67.7
XNOR_{AIL}	44.6M	57.7	74.7	46.0	40.0	66.8	47.7	65.6	63.2
OR_{AIL}	44.6M	**64.3**	**84.4**	49.5	**44.0**	**71.5**	47.1	**76.5**	**77.0**
OR, AND_{AIL} (d)	45.6M	57.5	74.7	45.6	41.3	68.3	44.9	64.5	63.0
OR, XNOR_{AIL} (p)	44.6M	59.8	80.5	45.8	41.4	67.2	48.2	67.5	68.1
OR, XNOR_{AIL} (d)	45.6M	62.8	83.7	49.1	43.3	68.1	**49.5**	73.8	72.2
OR, AND, XNOR_{AIL} (p)	44.6M	55.0	68.2	47.2	41.1	65.0	45.0	61.0	57.5

We found the network using OR_{AIL} activation function performed best overall, and across most of the subtasks. The second best performing activation functions were {OR_{AIL}, XNOR_{AIL} (d)} and ReLU.

A.19 Compositional Zero-Shot Learning

Zero-shot learning encompasses all problems which involve completing novel tasks which the subject has never seen before. The subject must infer both the task and its solution based on their previous experiences (a meta-learning task). Compositional zero-shot learning is a subset of zero-shot learning which involves combining knowledge about multiple stimulus properties together in novel pairings. For instance, if the network has been trained on "sliced bread," "sliced pear," and "caramelized pear," is it able to classify images of "caramelized bread" despite having never seen an example of this before?

We based our experiments on the Task-driven Modular Networks (TMN) proposed by Purushwalkam et al. (2019). We used the code shared by the authors, but were unable to replicate the results they reported in the paper when using a different random seed (see Table 15). We adapted this network by changing out all the ReLU activation functions in the gate and module networks with a different

activation function. Because the modules each terminate with an activation function, we needed to double the size of the hidden layer for some of our networks in order to maintain the dimensionality of the output. Consequently, some experiments had around 50% more parameters in total than others.

Experiments were performed on the MIT-States dataset (Isola et al., 2015). We trained and tested on the corresponding partitions of the dataset as introduced by Purushwalkam et al. (2019). We used the same paradigm as Purushwalkam et al. (2019): ADAM (Kingma & Ba, 2015) with a learning rate of 0.001 for the module network and 0.01 for the gating network, momentum 0.9, batch size 256, weight decay 5×10^{-5}. We evaluated the network using the AUC between seen and unseen samples (Purushwalkam et al., 2019). The network was trained until the validation AUC had plateaued, determined by not increasing for 5 epochs. We selected the model from the epoch with highest validation AUC to apply to the test set. The best performance was typically attained after around 5 epochs.

As shown in Table 15, we find that $\mathrm{OR}_{\mathrm{AIL}}$ performs best, but uses more parameters, and is very closely followed by $\{\mathrm{OR}, \mathrm{XNOR}_{\mathrm{AIL}}$ (d)$\}$,.

Table 15: Performance of TMN-based networks at compositional zero-shot learning (CZSL) on the MIT-States dataset. Mean (standard error) of $n=5$ random initializations. Bold: best. Underlined: top two. Background: linear color scale from ReLU baseline (white) to best (black).

			MIT-States Test AUC (%)		
Activation function	Mapping	Params	Top-1	Top-2	Top-3
TMN (Purushwalkam et al., 2019)	$1 \to 1$		2.9	7.1	11.5
TMN repro. (ReLU)	$1 \to 1$	438 k	2.47±0.07	6.42±0.09	10.27±0.11
Max	$2 \to 1$	650 k	2.42±0.07	6.37±0.08	10.28±0.06
Max, Min (d)	$1 \to 1$	438 k	2.53±0.06	6.69±0.11	10.61±0.14
$\mathrm{XNOR}_{\mathrm{AIL}}$	$2 \to 1$	650 k	1.22±0.05	3.47±0.12	5.82±0.19
$\mathrm{OR}_{\mathrm{AIL}}$	$2 \to 1$	650 k	**2.65**±0.05	**6.80**±0.08	**10.78**±0.13
$\mathrm{OR}, \mathrm{AND}_{\mathrm{AIL}}$ (d)	$2 \to 2$	438 k	2.61±0.05	6.73±0.05	10.77±0.12
$\mathrm{OR}, \mathrm{XNOR}_{\mathrm{AIL}}$ (d)	$2 \to 2$	438 k	1.89±0.13	5.12±0.21	8.35±0.24

Since we could not flexibly scale the total number of parameters in the network with the original architecture, we modified the TMN architecture by adding an additional linear layer to the end of each module which projects from the activation function to the embedding space. The modules would otherwise terminate with an activation function, which makes it difficult to handle activation functions which map $2 \to 1$. We dub the modified version of the network "TMNx."

Comparing the TMN results in Table 15 to TMNx in Table 16, we can see that adding the extra linear layer improved performance of the network in of itself. Intuitively, this makes sense since the output of the TMN modules are weighted with the output of the gating network and then summed, and this weighting and summing of evidence is best performed with on logits instead of the truncated output of ReLU units. But also, performance may have improved just because the model became larger.

We performed a hyperparameter search across the training parameters for the new network against the validation set using the ReLU activation function only. We adopted the hyperparameters discovered for ReLU for all other activation functions. The batch size was reduced to 128 due to the increase in size of the model. The discovered hyperparameters were a learning rate of 3×10^{-3} for both the module and gating network, and a weight decay of 1×10^{-5}. Other training hyperparameters, such as the ratio of negative samples to present, were left unchanged.

In order to match the number of parameters in the network, we used the original TMN hidden width of 64 for the module and gater networks with activation functions which map $1 \to 1$, and increased this to 70 for activation function which map $2 \to 1$, to maintain the total number of trainable parameters. To investigate a comprehensive set of baselines, we compared against every activation function implemented in PyTorch v1.10. The results are shown in Table 16.

We found that the ELU family of activations (ELU, CELU, SELU) performed best across all three top-k values. Our best activation functions were $\mathrm{OR}_{(\mathrm{N})\mathrm{IL}}$ and $\{\mathrm{OR}, \mathrm{AND}_{(\mathrm{N})\mathrm{IL}}$ (d)$\}$, which came next alongside Softplus and Tanh, outperforming ReLU. Activation functions using XNOR and our AIL approximations performed less well on this task, and were did not surpass ReLU. These results suggests this is a domain where logical activation functions are less well suited.

Table 16: Performance of TMNx networks at compositional zero-shot learning (CZSL) on the MIT-States dataset. Pre-activation width of 60, 64, 70, or 72 (depending on activation function, to keep the number of parameters approximately constant). Mean (standard error) of $n=5$ random initializations. Bold: best. Underlined: top two. Italic: no significant difference from best (two-sided Student's t-test, $p>0.05$). Background: color scale from second-worst in column to best, linear with accuracy value.

			MIT-States Test Accuracy (%)			
Activation function	Mapping	Width	Params	Top-1	Top-2	
ReLU	$1\to1$	64	1.15M	2.76±0.08	7.11±0.08	11.26±0.09
LeakyReLU	$1\to1$	64	1.15M	2.72±0.09	6.99±0.17	10.95±0.20
PReLU	$1\to1$	64	1.15M	2.83±0.06	7.25±0.14	11.44±0.17
Softplus	$1\to1$	64	1.15M	*2.92*±0.11	*7.46*±0.19	11.75±0.18
ELU	$1\to1$	64	1.15M	*3.13*±0.05	*7.78*±0.14	*12.04*±0.20
CELU	$1\to1$	64	1.15M	***3.17***±0.06	***7.81***±0.15	*12.00*±0.19
SELU	$1\to1$	64	1.15M	*3.05*±0.04	*7.77*±0.10	***12.37***±0.16
GELU	$1\to1$	64	1.15M	2.81±0.06	7.19±0.11	11.34±0.15
SiLU	$1\to1$	64	1.15M	2.87±0.08	*7.34*±0.14	11.60±0.18
Hardswish	$1\to1$	64	1.15M	2.84±0.08	7.23±0.11	11.48±0.11
Mish	$1\to1$	64	1.15M	2.90±0.08	*7.37*±0.17	11.59±0.13
Softsign	$1\to1$	64	1.15M	2.90±0.04	7.28±0.09	11.53±0.18
Tanh	$1\to1$	64	1.15M	2.93±0.09	*7.47*±0.16	11.64±0.20
GLU	$2\to1$	70	1.16M	2.59±0.06	6.86±0.15	10.91±0.24
Max	$2\to1$	70	1.16M	2.45±0.08	6.52±0.18	10.53±0.29
Max, Min (d)	$2\to2$	64	1.15M	2.53±0.10	6.61±0.15	10.65±0.15
SignedGeomean	$2\to1$	70	1.16M	1.65±0.04	4.40±0.08	7.18±0.07
XNOR_{IL}	$2\to1$	70	1.16M	2.03±0.04	5.43±0.02	8.82±0.05
OR_{IL}	$2\to1$	70	1.16M	2.90±0.05	*7.40*±0.11	11.68±0.14
OR, AND_{IL} (d)	$2\to2$	64	1.15M	2.68±0.06	7.05±0.08	11.17±0.12
OR, XNOR_{IL} (p)	$2\to1$	72	1.19M	2.41±0.09	6.20±0.13	9.82±0.14
OR, XNOR_{IL} (d)	$2\to2$	64	1.15M	2.60±0.04	6.68±0.12	10.71±0.17
OR, AND, XNOR_{IL} (p)	$2\to1$	72	1.19M	2.50±0.04	6.34±0.16	10.05±0.24
OR, AND, XNOR_{IL} (d)	$2\to3$	60	1.15M	2.39±0.05	6.31±0.07	10.17±0.12
XNOR_{NIL}	$2\to1$	70	1.16M	2.08±0.07	5.55±0.09	9.11±0.10
OR_{NIL}	$2\to1$	70	1.16M	2.92±0.07	*7.42*±0.13	11.74±0.14
OR, AND_{NIL} (d)	$2\to2$	64	1.15M	2.82±0.11	*7.43*±0.23	11.69±0.19
OR, XNOR_{NIL} (p)	$2\to1$	72	1.19M	2.50±0.08	6.51±0.14	10.40±0.17
OR, XNOR_{NIL} (d)	$2\to2$	64	1.15M	2.48±0.03	6.56±0.11	10.63±0.12
OR, AND, XNOR_{NIL} (p)	$2\to1$	72	1.19M	2.48±0.03	6.54±0.07	10.38±0.08
OR, AND, XNOR_{NIL} (d)	$2\to3$	60	1.15M	2.54±0.05	6.74±0.11	10.72±0.11
XNOR_{AIL}	$2\to1$	70	1.16M	1.88±0.06	4.94±0.15	7.99±0.19
OR_{AIL}	$2\to1$	70	1.16M	2.61±0.08	6.83±0.12	10.97±0.20
OR, AND_{AIL} (d)	$2\to2$	64	1.15M	2.81±0.06	7.12±0.10	11.27±0.14
OR, XNOR_{AIL} (p)	$2\to1$	72	1.19M	2.50±0.06	6.34±0.14	10.07±0.21
OR, XNOR_{AIL} (d)	$2\to2$	64	1.15M	2.46±0.04	6.51±0.11	10.45±0.17
OR, AND, XNOR_{AIL} (p)	$2\to1$	72	1.19M	2.45±0.12	6.33±0.21	10.09±0.23
OR, AND, XNOR_{AIL} (d)	$2\to3$	60	1.15M	2.57±0.03	6.83±0.11	10.97±0.17
$\text{XNOR}_{\text{NAIL}}$	$2\to1$	70	1.16M	1.78±0.04	4.84±0.11	7.95±0.19
OR_{NAIL}	$2\to1$	70	1.16M	2.51±0.03	6.58±0.12	10.56±0.24
OR, AND_{NAIL} (d)	$2\to2$	64	1.15M	2.67±0.04	6.98±0.06	11.02±0.16
OR, $\text{XNOR}_{\text{NAIL}}$ (p)	$2\to1$	72	1.19M	2.29±0.10	6.13±0.19	9.95±0.17
OR, $\text{XNOR}_{\text{NAIL}}$ (d)	$2\to2$	64	1.15M	2.43±0.06	6.32±0.09	10.28±0.18
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (p)	$2\to1$	72	1.19M	2.45±0.07	6.34±0.14	10.11±0.12
OR, AND, $\text{XNOR}_{\text{NAIL}}$ (d)	$2\to3$	60	1.15M	2.58±0.09	6.85±0.18	10.94±0.24