Mathematical Models of Computation in Superposition

Kaarel Hänni [*] [1] **Jake Mendel** [*] [2] **Dmitry Vaintrob** [*] **Lawrence Chan**

Abstract

Superposition – when a neural network represents more "features" than it has dimensions – seems to pose a serious challenge to mechanistically interpreting current AI systems. Existing theory work studies *representational* superposition, where superposition is only used when passing information through bottlenecks. In this work, we present mathematical models of *computation* in superposition, where superposition is actively helpful for efficiently accomplishing the task.

We first construct a task of efficiently emulating a circuit that takes the AND of the $\binom{m}{2}$ pairs of each of m features. We construct a 1-layer MLP that uses superposition to perform this task up to ε-error, where the network only requires $\tilde{O}(m^{\frac{2}{3}})$ neurons, even when the input features are *themselves in superposition*. We generalize this construction to arbitrary sparse boolean circuits of low depth, and then construct "error correction" layers that allow deep fully-connected networks of width d to emulate circuits of width $\tilde{O}(d^{1.5})$ and *any* polynomial depth. We conclude by providing some potential applications of our work for interpreting neural networks that implement computation in superposition.

1. Introduction

Mechanistic interpretability seeks to decipher the algorithms utilized by neural networks (Olah et al., 2017; Elhage et al., 2021; Räuker et al., 2023; Olah et al., 2020; Meng et al., 2023; Geiger et al., 2021; Wang et al., 2022; Conmy et al., 2024). A significant obstacle is that neurons are *polysemantic* – activating in response to various unrelated inputs (Fusi et al., 2016; Nguyen et al., 2016; Olah et al., 2017; Geva et al., 2021; Goh et al., 2021). As a proposed explanation for polysemanticity, Olah et al. (2020) introduce the 'superposition hypothesis' (see also Arora et al. (2018); Elhage et al. (2022)): the idea that networks represent many more concepts in their activation spaces than they have neurons by sparsely encoding features as nearly orthogonal directions.

Previous work has studied how networks can store more features than they have neurons in a range of toy models ((Elhage et al., 2022; Scherlis et al., 2022)). However, previous models of superposition either involve almost no computation (Elhage et al., 2022) or rely on some part of the computation not happening in superposition (Scherlis et al., 2022). Insofar as neural networks are incentivized to learn as many circuits as possible (Olah et al., 2020), they are likely to compute circuits in the most compressed way possible. Therefore, understanding how networks can undergo more general computation in a fully superpositional way is valuable for understanding the algorithms they learn.

In this paper, we lay the groundwork for understanding computation in superposition *in general*, by studying how neural networks can emulate sparse boolean circuits.

- In Section 2, we clarify existing definitions of linearly represented features, and propose our own definition which is more suited for reasoning about computation.
- In Section 3, we focus our study on the task of emulating the particular boolean circuit we call the *Universal AND* (U-AND) circuit. In this task, a neural network must take in a set of boolean features in superposition, and compute the pairwise logical ANDs of these features in a single layer with as few hidden neurons as possible. We present a construction which allows for many more new features to be computed than the number of hidden neurons, with outputs represented natively in superposition. We argue that real neural networks may well implement our construction in the wild by proving that randomly initialised networks are very likely to emulate U-AND.
- In Section 4 we demonstrate a second reason why this task is worth studying: it is possible to modify our construction to allow a wide range of large boolean circuits to be emulated entirely in superposition, provided that they satisfy a certain sparsity property.

We conclude with a discussion of the limitations of our for-

[*] Equal contribution, authors ordered alphabetically by last name. [1] Cadenza Labs / Caltech. [2] Apollo Research. Correspondence to: Lawrence Chan <chanlaw@berkeley.edu>.

Presented in the inaugural Mechanistic Interpretability Workshop, at the 41st International Conference on Machine Learning (2024.

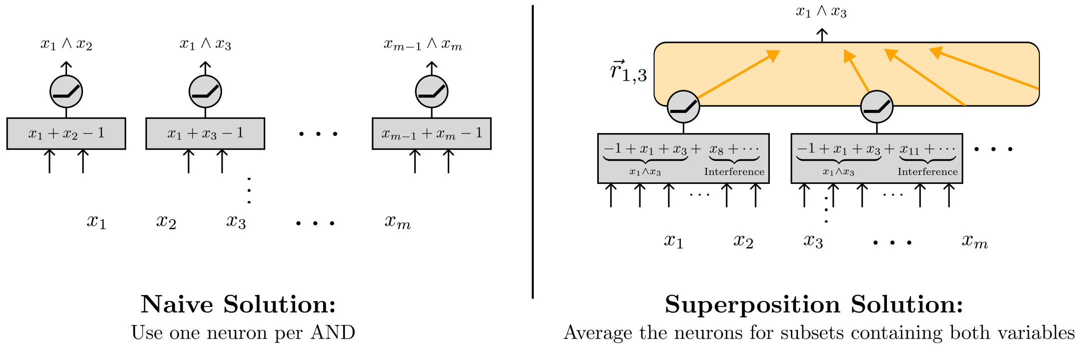

$x_1 \wedge x_2$
$x_1 \wedge x_3$
$x_{m-1} \wedge x_m$
$x_1 + x_2 - 1$
$x_1 + x_3 - 1$
$x_{m-1} + x_m - 1$
x_1
x_2
x_3
x_m
Naive Solution:
Use one neuron per AND
$x_1 \wedge x_3$
$\vec{r}_{1,3}$
$\underbrace{-1 + x_1 + x_3}_{x_1 \wedge x_3} + \underbrace{x_8 + \cdots}_{\text{Interference}}$
$\underbrace{-1 + x_1 + x_3}_{x_1 \wedge x_3} + \underbrace{x_{11} + \cdots}_{\text{Interference}}$
x_1
x_2
x_3
x_m
Superposition Solution:
Average the neurons for subsets containing both variables

Figure 1. The naive way to linearly represent the pairwise ANDs of m boolean variables using an MLP is to use one neuron to compute the AND of each pair of variables (left). This requires $\binom{m}{2} = O(m^2)$ neurons. However, when inputs are sparse, there is a much more efficient implementation using superposition (right). Here, each neuron checks for whether or not *at least two variables* are active in a subset of random variables. Then, for any pair of variables, we can read off the AND of that pair by averaging together the activations of all neurons corresponding to the subsets containing both variables. With appropriately chosen subsets, we can ε-linearly represent all pairwise ANDs using only $\tilde{O}(m^{\frac{2}{3}})$ neurons, even when the inputs are themselves represented in superposition (Section 3).

mal models, including the fact that our results are asymptotic and deal with only boolean features, and provide directions of future work that could address them.

2. Background and setup

2.1. Notation and conventions

Asymptotic complexity and $\tilde{O}$ notation We make extensive use of standard Bachmann–Landau ("big O") asymptotic notation. We use $\tilde{O}$ to indicate that we are ignoring polylogarithmic factors:

$$\tilde{O}(g(n)) := O(g(n) \log^k n) \quad \text{for some } k \in \mathbb{Z}.$$

(And so forth for $\tilde{\Theta}$, $\tilde{\Omega}$, etc.)

Fully connected neural networks We use $\mathcal{M}_w : X \to Y$ to denote a neural network model parameterized by w that takes input $x \in X$ and outputs $\mathcal{M}_w(x) \in Y$. In this work, we study fully-connected networks consisting of L MLP layers with ReLU activations:

$$\begin{aligned}
\vec{a}^{(0)}(x) &= x \\
\vec{a}^{(l)}(x) &= \mathrm{MLP}^{(l)}(\vec{a}^{(l-1)}(x)) \\
&= \mathrm{ReLU}(W_{\text{in}}{}^{(l)} \vec{a}^{(l-1)}(x) + w_{\text{bias}}{}^{(l)}) \\
\mathcal{M}_w(x) &= W_{\text{out}} \vec{a}^{(L)},
\end{aligned}$$

where $\mathrm{ReLU}(x) = \max(0, x)$ with max taken elementwise. We assume that our MLPs have width d for all hidden layers, that is, $\vec{a}^{(l)} \in \mathbb{R}^d$ for all $l \in \{1, ..., L\}$. For simplicity's sake we will be dropping l whenever we only talk about a single layer at a time.

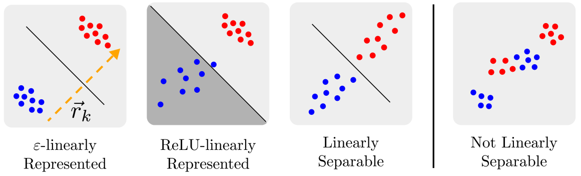


Figure 2. In Section 2.2, we distinguish between boolean features that are ε-*linearly represented* (left), ReLU-*linearly represented* (center left), and those that are only *linearly separable* (i.e. weakly linearly represented) (center right). Red/blue indicates the presence or absence of the feature. In addition to being linearly separable, ε-linearly represented features must satisfy the further condition that the variance in the readoff direction $\vec{r}_k$ *within* the positive and negative clusters is small compared to the margin between the two.

Features and feature vectors Following previous work in mechanistic interpretability (e.g. Tamkin et al. (2023); Rajamanoharan et al. (2024)), we suppose that the activations of a model can be thought of as representing $m > d$ boolean *features* $f_k : X \to \{0, 1\}$ of the input *in superposition*. That is,

$$\vec{a}(x) = \sum_{i=1}^{m} \vec{\phi}_k f_k(x)$$

for some set of *feature vectors* $\vec{\phi}_1, ..., \vec{\phi}_m \in \mathbb{R}^d$ and features $f_1, ..., f_m : X \to \{0, 1\}$. Equivalently,

$$\vec{a}^{(l)}(x) = \Phi \boldsymbol{b}$$

where $\Phi = (\vec{\phi}_1, ..., \vec{\phi}_m)$ is the $d \times m$ *feature encoding* matrix with columns equal to the feature vectors and $\boldsymbol{b} \in \{0, 1\}^m$ is the boolean vector with entries $\boldsymbol{b}_k = f_k(x)$.

In addition, as in previous work, we assume that these features are s-*sparse*, in that only at most $s \ll d, m$ features f_i can be nonzero for any input x (equivalently, $||\boldsymbol{b}||_1 \le s$.) For clarity, we preferentially use $k, \ell \in \{1, ..., m\}$ to index *features* (in $\{0,1\}^m$) and $i, j \in \{1, ..., d\}$ to index the standard neuron basis of activations (in $\mathbb{R}^d$).

Sparse boolean circuits We construct tasks where a neural network needs to emulate a boolean circuit $\mathcal{C}\colon \{0,1\}^m \to \{0,1\}^{m'}$. We assume that this circuit can be written as $\mathcal{C} = \mathcal{C}_L \circ \cdots \circ \mathcal{C}_1$, where each intermediate "layer" $\mathcal{C}_l : \{0,1\}^m \to \{0,1\}^m$ is a collection of m parallel boolean gates (of fan-in up to 2), for $l < L$. We say that a circuit $\mathcal{C}$ is s-sparse on boolean input $\boldsymbol{b} \in \{0,1\}^m$ if the input $\boldsymbol{b}^{(0)} = \boldsymbol{b}$ and all intermediate activations $\boldsymbol{b}^{(l)} = \mathcal{C}_i(\boldsymbol{b}^{(l-1)})$ are s-sparse, i.e. they satisfy $||\boldsymbol{b}^{(i)}||_1 \le s$.

2.2. Strong and weak linear representations

Given the activations of a neural network at a particular layer $a^{(l)}\colon X \to \mathbb{R}^d$, we can also ask what features are *linearly represented* by $a^{(l)}$. In this section, we present three definitions for a feature being linearly represented by $a^{(l)}$, which we illustrate in Figure 2.

The standard definition of linear representation is based on whether or not the representations of positive and negative examples can be separated by a hyperplane:

Definition 1 (Weak linear representations)**.** *We say that a binary feature* f_k *is* weakly linearly represented *by* $a\colon X \to \mathbb{R}^d$ *(or* linearly separable *in* a*) if there exists some* $\vec{r}_k \in \mathbb{R}^d$ *such that for all* $x_1, x_2 \in X$ *where* $f_k(x_1) = 0$ *and* $f_k(x_2) = 1$*, we have:*

$$\vec{r}_k \cdot a(x_1) < \vec{r}_k \cdot a(x_2).$$

Or, equivalently, the sets $\{x | f_k(x) = 0\}$ *and* $\{x | f_k(x) = 1\}$ *are separated by a hyperplane normal to* $\vec{r}_k$*.*

That being said, features being linearly separable does not mean a neural network can easily "make use" of the features. For some weakly linearly represented features f_1 and f_2, neither $f_1 \wedge f_2$ nor $f_2 \vee f_2$ need to be linearly represented, even if their read-off vectors $\vec{r}_1, \vec{r}_2$ are *orthogonal* (Figure 3). In fact, a stronger statement is true: it might not even be possible to linearly separate $f_1 \wedge f_2$ or $f_2 \vee f_2$ in MLP $\circ\, a$, that is, even after applying an MLP to the activations (see Theorem 9 in Appendix C.1).

As a result, in this paper we make use of a more restrictive notion of a feature being linearly represented:

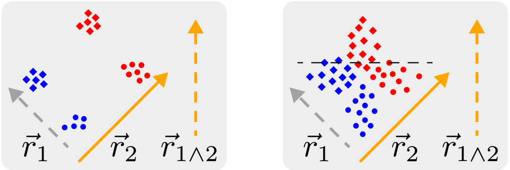
Definition 2 (ε-linear representations)**.** *Let* X *be a set of inputs and* $a\colon X \to \mathbb{R}^d$ *be the activations of a neural network (in a particular position/layer in a given model). We say that* $f_1, \ldots, f_m$ *are* linearly represented with interference ε *(or* ε-linearly represented *from these activation vectors) if there exists a* read-off matrix $\mathbf{R} \in \mathrm{Mat}_{m \times d}$ *with rows* $\vec{r}_1, \ldots, \vec{r}_m \in \mathbb{R}^d$ *such that for all* $k \in \{1, \ldots, m\}$ *and all* $x \in X$*, we have*

$$|\vec{r}_k \cdot \vec{a}(x) - f_k(x)| < \varepsilon.$$

We refer to $\vec{r}_k$ *as a* read-off vector *for the feature* f_k*. It follows that if* $\vec{a}(x) = \sum_{i=1}^m \vec{\phi}_k f_k(x)$*, then we have:*

$$||\mathbf{R}\Phi - \mathbf{Id}_m||_\infty < \varepsilon$$

where $\mathbf{Id}_m$ *is the* $m \times m$ *identity matrix*[1].



ε-linearly represented features compose

Linearly-separable features do not compose

Figure 3. When two features f_1, f_2 are ε-linearly represented in activations $a(x)$, we can use two MLP neurons with input weights $\vec{r}_1, \vec{r}_2$ to read-off the two features, after which $f_1 \wedge f_2$ and $f_1 \vee f_2$ are ε-linearly represented in the MLP activations $\mathrm{MLP}(a(x))$. However, because linearly-separable features can have arbitrarily small margin, there might exist *no* MLP such that $f_1 \wedge f_2$ and $f_1 \vee f_2$ are linearly separable in $\mathrm{MLP}(a(x))$.

For brevity's sake, we very slightly abuse notation here to include the bias term in $\vec{r}_k$. This is equivalent to assuming that one of $\vec{a}$'s outputs is a constant, that is, $a_i(x) = c$ for all x for some $i \in \{1, ..., d\}$ and some $c \in \mathbb{R}$.

In contrast to features that are merely linearly separable, features that are ε-linearly represented are easy to linearly separate, as we show in Figure 3. We formalize and prove this in Theorem 10 in Appendix C.1.

Comparison with Anthropic's Toy Model of Superposition Finally, Elhage et al. (2022) and Bricken et al. (2023) consider a definition of linearly represented feature that involves using a ReLU to remove negative interference:

Definition 3 (ReLU-linear representations)**.** *A set of* m *binary features* $\vec{F} = (f_1, ..., f_m)$ *is ReLU-linearly represented in* $a\colon X \to \mathbb{R}^d$ *with error* ε *if there exists a read-off matrix* $\mathbf{R} \in \mathrm{Mat}_{m \times d}$ *such that*

$$\mathbb{E}_{x \in X} ||\vec{F}(x) - \mathrm{ReLU}\left(\mathbf{R}a(x)\right)||_2 < \varepsilon.$$

[1] In some cases if the feature vectors satisfy $|\Phi^T \Phi - \mathbf{Id}_m| \le \mu$ — that is, if the feature vectors are *almost orthogonal with interference* μ, then the features vectors can function as their own readoffs.

Note that in contrast to ε-linearly represented features, where each individual feature must be able to be read off using an affine function with small error on *every* datapoint, ReLU-linear representated features are read off using a MLP layer with m neurons (one per feature), such that the *expected* ℓ_2 loss (summed across all m features) is small.

3. Universal ANDs: a model of single-layer MLP superposition

We start by presenting one of the simplest non-trivial boolean circuits: namely, the one-layer circuit that computes the pairwise AND of the input features. Note that due to space limitations, we include only proof sketches in the main body and may ignore some regularity conditions in the theorem statement. See Appendix E for more rigorous theorem statements and proofs.

Definition 4 (The universal AND boolean circuit)**.** *Let $\boldsymbol{b} \in \{0,1\}^m$ be a boolean vector. The universal AND (or U-AND) circuit has m inputs and $\binom{m}{2}$ outputs indexed by unordered pairs k, ℓ of locations and is defined by*

$$\mathcal{C}_{\text{UAND}}(\boldsymbol{b})_{k,\ell} := \boldsymbol{b}_k \wedge \boldsymbol{b}_\ell.$$

In other words, we apply the AND gate to all possible pairs of distinct inputs to produce $\binom{m}{2}$ outputs.

We will build our theory of computation starting from a single-layer neural net that emulates the universal AND when the input $\boldsymbol{b}$ is s-sparse for some $s \in \mathbb{N}$ (this implies that the output has sparsity $O(s^2)$).

3.1. Superposition in MLP activations enables more efficient U-AND

First, consider the naive implementation, where we use one ReLU to implement each AND using the fact that for boolean x_1, x_2:

$$\text{ReLU}(x_1 + x_2 - 1) = x_1 \wedge x_2.$$

This requires $\binom{n}{2} = O(n^2)$ neurons, each of which is monosemantic in that it represents a single natural feature. In contrast, by using sparsity, we can construct using exponentially fewer neurons (Figure 1):

Theorem 1 (U-AND with basis-aligned inputs)**.** *Fix a sparsity parameter $s \in \mathbb{N}$. Then for large input length m, there exists a single-layer neural network $\mathcal{M}_w(x) = \text{MLP}(x) = \text{ReLU}(W_{in}x + w_{bias})$ that ε-linearly represents the universal AND circuit $\mathcal{C}_{\text{UAND}}$ on s-sparse inputs, with width $d = \tilde{O}_m(1/\varepsilon^2)$ (i.e. polylogarithmic in m).*

Proof. (sketch) To show this, we construct an MLP such that each neuron checks whether or not at least two inputs in a small random subset of the boolean input $\boldsymbol{b}$ are active (see also Figure 1). Intuitively, since the inputs are sparse, each neuron can be thought of as checking the ANDs of any pair of input variables $\boldsymbol{b}_{k_1}, \boldsymbol{b}_{k_2}$ in the subset, with interference terms corresponding to all the other variables. That is, we can write the preactivation of each neuron as the sum of the AND of $\boldsymbol{b}_{k_1}, \boldsymbol{b}_{k_2}$ and some interference terms:

$$\underbrace{-1 + \boldsymbol{b}_{k_1} + \boldsymbol{b}_{k_2}}_{\boldsymbol{b}_{k_1} \wedge \boldsymbol{b}_{k_2}} + \underbrace{\sum_{k' \neq k_1, k_2} \boldsymbol{b}_{k'}}_{\text{interference terms}}$$

We then use the sparsity of inputs to bound the size of the interference terms, and show that we can "read-off" the AND of $\boldsymbol{b}_{k_1}, \boldsymbol{b}_{k_2}$ by averaging together the value of post-ReLU activations of the neurons connected to $\boldsymbol{b}_{k_1}, \boldsymbol{b}_{k_2}$. We then argue that this averaging reduces the size of the interference terms to below ε.

Specifically, we construct input weights $W_{\text{in}} \in \text{Mat}_{d \times m}$ such that the input to each neuron is connected to the kth entry of the input $\boldsymbol{b}_k$ with weight 1 with probability $p = \log^2 m/\sqrt{d}$, and weight 0 otherwise. We set the bias of each neuron to -1.

Let $\Gamma(k)$ be indices of neurons that have input weight 1 for $\boldsymbol{b}_k$, and $\Gamma(k_1, k_2)$ be the indices of neurons that have input weight 1 for $\boldsymbol{b}_{k_1}, \boldsymbol{b}_{k_2}$, $\Gamma(k_1, k_2, k_3)$ be the indices of neurons reading from all of $\boldsymbol{b}_{k_1}, \boldsymbol{b}_{k_2}, \boldsymbol{b}_{k_3}$, and so forth. By construction, $\Gamma(k_1)$ has expected size $\Theta(\log^2 m \cdot \sqrt{d})$, $\Gamma(k_1, k_2)$ has expected size $\Theta(\log^4 m)$, and $\Gamma(k_1, k_2, k_3)$ has expected size $\Theta(\log^6 m/\sqrt{d})$. In general, the set of indices for n such inputs has expected size $\Theta(\log^{2n} /d^{(n/2-1)})$

Our read-off vector $\vec{r}$ for the AND $\boldsymbol{b}_{k_1} \wedge \boldsymbol{b}_{k_2}$ will have entries:

$$\vec{r}_{(i)} = \begin{cases} \frac{1}{|\Gamma(k_1,k_2)|} & i \in |\Gamma(k_1, k_2)| \\ 0 & \text{otherwise} \end{cases}$$

We then check that $\vec{r} \cdot \text{MLP}(\boldsymbol{b})$ gives the correct output in each of three cases. Note that for any input, $\vec{r} \cdot \text{MLP}(\boldsymbol{b}) \geq \boldsymbol{b}_{k_1} \wedge \boldsymbol{b}_{k_2}$, so it suffices to upper bound the average number of non-k_1, k_2 inputs that are non-zero, divided by the total number of neurons in $\Gamma(k_1, k_2)$.

- When $\boldsymbol{b}_{k_1} = \boldsymbol{b}_{k_2} = 0$, the interference terms in each read-off neuron have value at most s, and there are at most

  $$\sum_{\boldsymbol{b}_{k'} = \boldsymbol{b}_{k''} = 1} |\Gamma(k_1, k_2, k', k'')| = \Theta(s^2 \cdot \log^8 m/d)$$

  such neurons outputting non-zero values. So the error is bounded above by

  $$\frac{s \cdot \sum_{k' \neq k_1, k_2} |\Gamma(k_1, k_2, k', k'')|}{|\Gamma(k_1, k_2)|} = \Theta(s^3 \cdot \log^4 m/d).$$

- When $\boldsymbol{b}_{k_1} = 1$ or $\boldsymbol{b}_{k_2} = 1$, the interference terms in each read-off neuron have value at most $s - 2$, and there are at most

  $$\sum_{\boldsymbol{b}_{k'}=1} |\Gamma(k_1, k_2, k')| = \Theta(s \cdot \log^6 m/\sqrt{d})$$

  neurons that have such interference terms.

  So the error is bounded above by

  $$\frac{s-2}{|\Gamma(k_1, k_2)|} \sum_{k' \neq k_1, k_2} |\Gamma(k_1, k_2, k')| \\ = \Theta(s^2 \cdot \log^2 m/\sqrt{d})$$

Combining the above, we get that the read-off error is $O(\log^4 m/\sqrt{d})$, and so setting $d = \Theta(\log^8 m/\varepsilon^2) = \tilde{O}_m(1/\varepsilon^2)$ gives us an error that is $< \varepsilon$ outside negligible probability.

□

3.2. Neural networks can implement efficient U-AND even with inputs in superposition

Note that in Theorem 1, we assume that the network gets m basis-aligned inputs (that is, not in superposition). However, it turns out that we can extend the result in Theorem 1 to inputs in superposition.

Theorem 2 (U-AND with inputs in superposition)**.** *Let $s \in \mathbb{N}$ be a fixed sparsity limit and $\varepsilon < 1$ a fixed interference parameter. There exists a feature encoding Φ and single-layer neural net $\mathcal{M}_w(x) = \mathrm{MLP}(x) = \mathrm{ReLU}(W_{in}x + w_{bias})$ with input size and width $d = \tilde{O}(\sqrt{m}/\varepsilon^2)$, where $\mathcal{M}_w \circ \Phi$ ε-linearly represents $\mathcal{C}_{\mathrm{UAND}}$ on all s-sparse inputs $\boldsymbol{b}$.*

Proof. (sketch) By picking almost orthogonal unit-norm vectors $\Phi = (\vec{\phi}_1, \ldots, \vec{\phi}_m)$, we can recover each feature up to error ε using readoffs $\mathbf{R} = \Phi^T$. Take the input weight $W_{\text{in}} \in \mathrm{Mat}_{d\times m}$ for the MLP constructed in the proof of Theorem 1. Using $W_{\text{in}}' = W_{\text{in}}\mathbf{R}$ and $w_{\text{bias}}' = w_{\text{bias}}$ suffices, as this gives us

$$\begin{aligned}\mathcal{M}_w \circ \Phi(\boldsymbol{b}) &= \mathrm{ReLU}(W_{\text{in}}\mathbf{R}\Phi\boldsymbol{b} + w_{\text{bias}}) \\ &\approx \mathrm{ReLU}(W_{\text{in}}\boldsymbol{b} + w_{\text{bias}}),\end{aligned}$$

which is just the model from Theorem 1, which ε-linearly represents $\mathcal{C}_{\mathrm{UAND}}$ as desired. Carefully tracking error terms shows that we need $d = \tilde{\Theta}(\sqrt{m})$ neurons. □

3.3. Randomly initialized neural networks linearly represent U-AND

While the results in previous section show that there exist *some* network weights that ε-linearly represents the U-AND circuit $\mathcal{C}_{\mathrm{UAND}}$, there still is a question of whether neural networks can learn to represent many ANDs starting from the standard initialization. In this section, we provide some theoretical evidence – namely, that sufficiently wide *randomly initialized* one-layer MLPs ε-linearly represent $\mathcal{C}_{\mathrm{UAND}}$.

Theorem 3 (Randomly initialized MLPs linearly represent U-AND)**.** *Let $\mathrm{MLP} : \mathbb{R}^m \to \mathbb{R}^d$ be a one-layer MLP with $d = \tilde{\Omega}(1/\varepsilon^2)$ neurons that takes input **b**, and where W_{in} is drawn i.i.d from a normal distribution $\mathcal{N}(0, \delta^2)$ and $w_{bias} = \vec{0}$. Then this MLP ε-linearly represents $\mathcal{C}_{\mathrm{UAND}}$ on s-sparse inputs outside of negligible probability.*

Proof. (Sketch) We prove this by constructing a read-off vector $\vec{r}$ for each pair of features k_1, k_2. Let σ be the sign function

$$\sigma(x) = \begin{cases} +1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}$$

and let $w_{i,k}$ be the contribution to the preactivation of neuron i from $\boldsymbol{b}_k$.

We construct $\vec{r}$ coordinatewise (that is, neuron-by-neuron). In particular, we set the ith coordinate of $\vec{r}$ to be

$$\vec{r}_i = \eta_i \left(\mathbf{1}_{\sigma(w_{i,k_1}) = \sigma(w_{i,k_2})} - \mathbf{1}_{\sigma(w_{i,k_1}) \neq \sigma(w_{i,k_2})} \right).$$

That is, if k_1 and k_2 contribute to the neuron preactivations with the same sign, then $\vec{r}_i = \eta_i$, else, $\vec{r}_i = -\eta_i$. Here, η_i is a scaling parameter of size $\Theta(\sqrt{s}/d)$ used to scale the read-off to be 1 when $\boldsymbol{b}_{k_1} = \boldsymbol{b}_{k_2} = 1$

When $\boldsymbol{b}_{k_1} = 0$ or $\boldsymbol{b}_{k_2} = 0$, the expected value of $\vec{r} \cdot \mathcal{M}_w(\boldsymbol{b})$ is zero, while the error terms have size $\tilde{O}_m(1/\sqrt{d})$. So setting $d = \tilde{\Omega}(1/\varepsilon^2)$ suffices to get error below ε with high probability.

When $\boldsymbol{b}_{k_1} = \boldsymbol{b}_{k_2} = 1$, the contribution from each neuron i to $\vec{r} \cdot \mathcal{M}_w(\boldsymbol{b})$ with $\sigma(w_{i,k_1}) = \sigma(w_{i,k_2})$ will be in expectation larger than those with $\sigma(w_{i,k_1}) \neq \sigma(w_{i,k_2})$ (as the standard deviation of the sum of two weights with equal signs is larger than the sum of two weights with different signs, and we apply a ReLU). By setting η to be the reciprocal of the difference in expected contributions, we have that this value has expectation 1. Again, as the error terms have size $\tilde{O}_m(1/\sqrt{d})$, it follows that setting $d = \tilde{\Omega}(1/\varepsilon^2)$ suffices to get error below ε with high probability, as desired.

□

Before proceeding, we record a corollary, which underscores the surprisingly strong asymptotic representability of the universal AND circuit.

Corollary 4. *For any fixed input size s, dimension d and $m = d^{O(1)}$ polynomial in d, there exists a "universal AND" model with hidden dimension d,*

$$\mathcal{M}_w : x \mapsto \mathrm{ReLU}(W_{in}(x))$$

from $\mathbb{R}^d$ to $\mathbb{R}^d$ and a feature matrix $\Phi \in \mathrm{Mat}_{m\times d}$ such that for any input $\boldsymbol{b}$ with sparsity $||\boldsymbol{b}||_1 = s$, we have that $\mathcal{M}_w(\Phi(\boldsymbol{b})) \in \mathbb{R}^d$ strongly linearly represents uAND($\boldsymbol{b}$) $\in \{0,1\}^{\binom{m}{2}}$ (with error at worst $\varepsilon = \tilde{O}(\frac{1}{\sqrt{d}})$).

4. MLPs as representing sparse boolean circuits

In the previous section we showed variants of computation in superposition at a single layer, for one of the simplest non-trivial boolean circuits. In this section, we extend these results to show that neural networks can efficiently represent *arbitrary* sparse boolean circuits.

As in Section 3, we include only proof sketches in the main body due to space limitations, and may also ignore some regularity conditions in our theorem statements. See Appendix E for more rigorous theorem statements and proofs.

4.1. Boolean circuits in single layer MLPs

We start by extending these results from Section 3 to ANDs of more than two variables.

Let $\mathcal{C}^{(n)}_{\mathrm{UAND}}$ be the boolean circuit of depth $L = \log(n)$ that computes the ANDs of each n-tuple of elements in $\boldsymbol{b}$.[2]

Lemma 5 ("High fan-in" U-AND)**.** *For each $n \in \mathbb{N}$, there exists a one-layer neural network $\mathcal{M}_w = \mathrm{MLP} : \mathbb{R}^m \to \mathbb{R}^d$ with width $d = \tilde{O}(n/\varepsilon^2)$ such that $\mathcal{M}_w(\boldsymbol{b})$ ε-linearly represents $\mathcal{C}^{(n)}_{\mathrm{UAND}}$ on s-sparse inputs.*

Proof. (sketch) We can extend the construction in the proof of Theorem 1 to allow for ANDs of exactly n variables, by considering index sets $\mathbf{I}(k_1, k_2, ..., k_n)$ of n variables, and changing the bias of each neuron from -1 to $-n + 1$. The expected size of an index set of n variables is $\mathbb{E}[|\mathbf{I}(k_1, k_2, ..., k_n)|] = p^n d$, and we require this expected value to be $\Omega(\log^4 m)$ to ensure that the index set is non-empty outside negligible probability (using the normal Chernoff and Union bounds). Therefore, we have to scale up the probability that any given value in W_{in} is 1: $p = \frac{\log^2 m}{d^{1/n}}$ suffices. A similar argument to the one found in the proof of Theorem 1 shows that all the interference terms are $o(1)$. □

As illustrated in Figure 4, Lemma 5 allows us to construct MLPs that ε-linearly represents arbitrary small circuits:

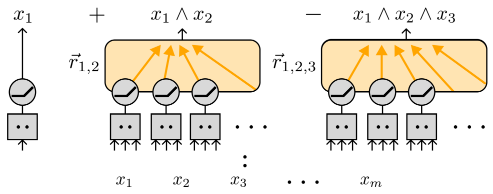


Figure 4. As discussed in Section 4.1, our U-AND construction can be extended to allow for arbitrarily high fan-in ANDs, which in turn allows for single-layer MLPs that linearly represent all small boolean circuits.

Theorem 6. *For any s-sparse circuit $\mathcal{C}$ of width m and depth L, there exists a feature encoding $\Phi \in \mathrm{Mat}_{d\times m}$ and a single-layer neural network $\mathcal{M}_w(x) = \mathrm{ReLU}(W_{in}x + w_{bias})$ of width $d = \tilde{O}(\sqrt{m})$ such that $\mathcal{M}_w(\Phi\boldsymbol{b})$ ε-linearly represents $\mathcal{C}(\boldsymbol{b})_k$ for all $k \in \{1, ..., m\}$ for some $\varepsilon = \tilde{O}(m^{-1/3})$.*

Proof. (sketch) First, apply the construction in Theorem 2 to show that there exists one-layer MLPs of width $d = \tilde{O}(\sqrt{m})$ that compute $\mathcal{C}^{(n)}_{\mathrm{UAND}}$ when the inputs are in superposition, where $n \in \{2, 3, ..., 2^L\}$.

Next, concatenate together the $2^L - 1$ networks of width $d = \tilde{O}(\sqrt{m})$ that ε-linearly represent each $\mathcal{C}^{(n)}_{\mathrm{UAND}}$ for $n \in \{2, 3, ..., 2^L\}$, when the inputs are in superposition. Since the output of *any* boolean circuits of depth L can be written as a linear combinations of ANDs of maximum fan-in 2^L, it follows that the concatenated network ε'-linearly represents any boolean circuit of depth L, for some ε' dependent on how many ANDs need to be added together to compute the circuit, as desired. □

4.2. Efficient boolean circuits via deep MLPs

The one-layer MLP in Theorem 6 has width that is exponential in the depth of the circuit. However, by combining pairwise U-AND layers (which linearly represent any one-layer boolean circuit) with "error correction" layers, we can construct deeper neural networks with sublinear width and depth linear in the depth of the circuit.

Lemma 7. *Assume that $m = \tilde{O}(d^{1.5})$, and c is some large polylog constant. Then for sufficiently small input interference $\varepsilon = \tilde{O}(1/\sqrt{d})$ there exists a 1-layer MLP $\mathcal{M}_w : \mathbb{R}^d \to \mathbb{R}^d$ that takes as input a boolean vector of length m encoded in d-dimensions using superposition and returns (outside negligible probability) an encoding of the same boolean vector with interference ε/c.*

Proof. See Theorem 22 in Appendix E.4. □

[2] Note that by our definition, boolean circuits are made of gates of fan-in at most 2. So computing the ANDs of n variables requires a boolean circuit of depth $\log(n)$.

By alternating between such "error correction" layers and U-AND layers, we can construct more efficient circuits:

Theorem 8. *Let $\mathcal{C} : \{0,1\}^m \to \{0,1\}^m$ be a circuit of width m and of depth $L = O(m^c)$ polynomial in m. There exists a neural network of width $d = \tilde{O}(m^{\frac{2}{3}} s^2)$ and with depth $2L$ such that $\mathcal{M}_w(\Phi \boldsymbol{b})$ ε-linearly $\mathcal{C}(\boldsymbol{b})$ for all but a negligible fraction of inputs $\boldsymbol{b}$ on which $\mathcal{C}$ is s-sparse.*

Proof. (sketch) As a single MLP layer can ε-linearly represent the ANDs of all input features (by Theorem 2), we can use one MLP layer to approximate each layer of the circuit. However, the naive construction suffers from (potentially) exponentially growing error. To fix this, we insert an error correction layer from Lemma 7 between every such layer. □

5. Related Work

The idea that neural networks could or should make use of distributed or compositional representations has been a mainstay of early neural network research (Rosenblatt, 1961; Holyoak, 1987; Fodor & Pylyshyn, 1988). Arora et al. (2018) were the first in the modern deep learning context to discuss that neural networks could store many features in superposition. Olah et al. (2020) developed this idea into the 'superposition hypothesis': the conjecture that networks use the same neurons for multiple circuits to maximise the number of circuits they can learn.

Many of our results are similar in flavor to those from the fields of sparse dictionary (Tillmann, 2014) and hyperdimensional computing (Zou et al., 2021), as all rely on useful properties of high-dimensional spaces. In addition, many of our boolean circuit results on randomly-initialized MLP layers are similar in flavor to universality results on randomly initialized neural networks with different non-linearities (Rahimi & Recht, 2008a;b). However, these results consider cases where there are fewer "true features" than there are dimensions, while the superposition hypothesis requires that the number of "true features" exceeds the dimensionality of the space. Randomized numerical linear algebra (Murray et al., 2023) studies the use of random projections to perform efficient computation, but in the context of reducing the cost of linear algebra operations such as linear regression or SVD with inputs and outputs represented in an axis-aligned fashion.

Superposition has been studied in a range of idealised settings: Elhage et al. (2022) provided the first examples of toy models which employed superposition to achieve low loss and Henighan et al. (2023) further explored superposition in a toy memorisation task. Notably, they study features that are ReLU-linear represented. (See Section 2.2 for more discussion.) Scherlis et al. (2022) study a model of using a small number of neurons with *quadratic* activations to approximately compute degree two polynomials. The models studied in all of these papers require sparse features of *declining* importance. In contrast, our model allows for sparse features that are equally important. More importantly, none of these listed works study performing computation with *inputs in superposition*.

Several papers have also explored the prevalence of superposition in language models. Gurnee et al. (2023) found that some bigrams are represented on sparse sets of neurons but not on any individual neurons. There is also a growing literature on using sparse dictionary learning to identify features in language models inspired by the superposition hypothesis (Cunningham et al., 2023; Bricken et al., 2023; Tamkin et al., 2023; Bloom, 2024; Braun et al., 2024; Templeton et al., 2024) although it is unclear how much evidence the success of sparse dictionary learning in finding human-interpretable features provides for the superposition hypothesis.

6. Discussion

6.1. Summary

In this work, we have presented a mathematical framework for understanding how neural networks can perform computation in superposition, where the number of features computed can greatly exceed the number of neurons. We have demonstrated this capability through the construction of a neural network that efficiently emulates the Universal AND circuit, computing all pairwise logical ANDs of input features using far fewer neurons than the number of output features. Furthermore, we have shown how this construction can be generalized to emulate a wide range of sparse, low-depth boolean circuits entirely in superposition. This work lays the foundation for a deeper understanding of how neural networks can efficiently represent and manipulate information, and highlights the importance of considering computation in superposition when interpreting the algorithms learned by these systems.

6.2. Practical Takeaways for Mechanistic Interpretability

Our primary motivation for undertaking this work was to glean insights about the computation implemented by neural networks. While we provide more potential takeaways in Appendix B, here we discuss what we think are two salient takeaways for interpretability:

Unused features The implementation of U-AND by random matrices (Theorem 3) suggests that certain concepts may be detectable through linear probes in a network's activation space without being actively utilized in subsequent computations. This phenomenon could explain the findings of Marks (2024), who observed that arbitrary XORs of concepts can be successfully probed in language models. Furthermore, it implies that successfully probing for a concept and identifying a direction that explains a high percentage of variance (e.g., 80%) may not constitute strong evidence of the model's actual use of that concept. Consequently, there is reason to be cautious about how many of the features identified by Sparse Autoencoders (Cunningham et al., 2023; Bricken et al., 2023; Bloom, 2024; Templeton et al., 2024) are actively employed by the model in its computation.

Robustness to noise This research underscores the critical role of error correction in networks performing computations in superposition. Effective error correction mechanisms should enable networks to rectify minor perturbations in their activation states, resulting in a nonlinear response in output when activation vectors are slightly altered along specific directions. Expanding on this concept, Heimersheim & Mendel (2023) conducted follow-up investigations, revealing the presence of *plateaus* surrounding activation vectors in GPT2-small (Radford et al., 2019). Within these plateaus, model outputs exhibit minimal variation despite small changes in activation values, providing weak evidence for an error correcting mechanism in the model's computation.

6.3. Limitations and future work

That being said, there are a number of ways in which the computational framework presented in this work is very likely to miss the full richness of computation happening in any given real neural network.

Firstly, this work studies computation on binary features. It is plausible that other kinds of features – in particular, discrete features which take on more than 2 distinct values, or continuous-valued features – occur commonly in real neural networks. It would be valuable to extend the understanding developed in this work to such non-binary features.

Secondly, though we do not require features to have declining importance, we do require features to be sparse, with each data point only having a small number of active features. It is plausible that not all features are sparse in practice (given the present state of empirical evidence, it even appears open to us whether a significant fraction of features are sparse in practice) – for instance, perhaps real neural networks partly use more compositional representations with dense features.

Thirdly, in this work, we have made a particular choice regarding what it takes for a feature to be provided in the input and to have been computed in the output: ε-linear representation (Definition 2). Future empirical results or theoretical arguments could call for revising this choice — for instance, perhaps an eventual full reverse-engineering picture would permit certain kinds of non-linear features.

Finally and least specifically, the way of looking at neural net computation suggested in this work could turn out to be thoroughly confused. We consider there to be a lot of room for the development of a more principled and empirically grounded picture.

Impact Statement

The primary impact of our work is to advance the field of mechanistic interpretability. While advancing this field may have many potential societal impacts, we feel that there are no direct, non-standard impacts of our work that are worth highlighting.

References

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Bellare. A note on negligible functions. *Journal of Cryptology*, 15:271–284, 2002.

Bloom, J. Open source sparse autoencoders for all residual stream layers of GPT2 small. https://www.alignmentforum.org/posts/f9EgfLSurAiqRJySD/, 2024.

Braun, D., Taylor, J., Goldowsky-Dill, N., and Sharkey, L. Identifying functionally important features with end-to-end sparse dictionary learning. *arXiv preprint arXiv:2405.12241*, 2024.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36, 2024.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Fodor, J. A. and Pylyshyn, Z. W. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2): 3–71, 1988.

Fusi, S., Miller, E. K., and Rigotti, M. Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66–74, 2016. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2016.01.010. URL https://www.sciencedirect.com/science/article/pii/S0959438816000118. Neurobiology of cognitive behavior.

Geiger, A., Lu, H., Icard, T., and Potts, C. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586, 2021.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories, September 2021. URL http://arxiv.org/abs/2012.14913. arXiv:2012.14913 [cs].

Goh, G., †, N. C., †, C. V., Carter, S., Petrov, M., Schubert, L., Radford, A., and Olah, C. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi: 10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Heimersheim, S. and Mendel, J. Interim research report: Activation plateaus and sensitive periods in transformer training, 2023. URL https://www.alignmentforum.org/posts/LajDyGyiyX8DNNsuF/interim-research-report-activation-plateaus-and-sensitive-1. Accessed: 2024-07-27.

Henighan, T., Carter, S., Hume, T., Elhage, N., Lasenby, R., Fort, S., Schiefer, N., and Olah, C. Superposition, memorization, and double descent. *Transformer Circuits Thread*, 2023.

Holyoak, K. J. Parallel distributed processing: explorations in the microstructure of cognition. *Science*, 236:992–997, 1987.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Marks, S. What's up with llms representing xors of arbitrary features?, 2024. URL https://www.alignmentforum.org/posts/hjJXCn9GsskysDceS/what-s-up-with-llms-representing-xors-of-arbitrar Accessed: 2024-07-27.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt, 2023.

Murray, R., Demmel, J., Mahoney, M. W., Erichson, N. B., Melnichenko, M., Malik, O. A., Grigori, L., Luszczek, P., Dereziński, M., Lopes, M. E., et al. Randomized numerical linear algebra: A perspective on the field with an eye to software. *arXiv preprint arXiv:2302.11474*, 2023.

Nguyen, A., Yosinski, J., and Clune, J. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks, 2016.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rahimi, A. and Recht, B. Uniform approximation of functions with random bases. In *2008 46th annual allerton conference on communication, control, and computing*, pp. 555–561. IEEE, 2008a.

Rahimi, A. and Recht, B. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 2008b.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Rosenblatt, F. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Räuker, T., Ho, A., Casper, S., and Hadfield-Menell, D. Toward transparent AI: A survey on interpreting the inner structures of deep neural networks, 2023.

Scherlis, A., Sachan, K., Jermyn, A. S., Benton, J., and Shlegeris, B. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Taggart, G. M. ProLU: A nonlinearity for sparse autoencoders. https://www.alignmentforum.org/posts/HEpufTdakGTTKgoYF/prolu-a-nonlinearity-for-sparse-autoencoders, 2024.

Tamkin, A., Taufeeque, M., and Goodman, N. D. Codebook features: Sparse and discrete interpretability for neural networks. *arXiv preprint arXiv:2310.17230*, 2023.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Tillmann, A. M. On the computational intractability of exact and approximate dictionary learning. *IEEE Signal Processing Letters*, 22(1):45–49, 2014.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Zou, Z., Alimohamadi, H., Imani, F., Kim, Y., and Imani, M. Spiking hyperdimensional network: Neuromorphic models integrated with memory-inspired framework. *arXiv preprint arXiv:2110.00214*, 2021.

A. Mathematical definitions

Here, we list and define the mathematical terms that we use throughout this work.

X	set of inputs
Y	set of outputs
$\mathcal{M}_w : X \to Y$	neural network with ReLU activations, parameterized by w
$\vec{a}^{(l)}(x) \in \mathbb{R}^d$	the activations of a neural network at layer l, $l \in \{0, ..., L\}$
$\mathrm{MLP}^{(l)} : \mathbb{R}^d \to \mathbb{R}^d$	the lth MLP layer, $\mathrm{MLP}^{(l)}(x) = \mathrm{ReLU}({W_{\text{in}}}^{(l)} x + {w_{\text{bias}}}^{(l)})$
$f_k : X \to \{0, 1\}$	boolean feature of the input, $k = 1, \dots, m$
$F : X \to \{0, 1\}^m$	the concatenation of m boolean features
$\vec{\phi}_k \in \mathbb{R}^d$	vector linearly representing the kth boolean feature
$\Phi \in \mathbb{R}^{d \times m}$	the feature embedding matrix, $\Phi = (\vec{\phi}_1, ..., \vec{\phi}_m)$
$\boldsymbol{b} = \boldsymbol{b}(x) \in \{0, 1\}^m$	a boolean vector of length m associated to an input/activation
$\boldsymbol{b}_k = \boldsymbol{b}_k(x) \in \{0, 1\}$	the kth entry in the boolean vector, equal to $f_k(x)$
$\lvert\lvert \boldsymbol{b}(x) \rvert\rvert_1$	"sparsity", a.k.a. number of bits that are "on" for the boolean vector $\boldsymbol{b}$, equal to $\sum_{k=1}^{m} f_k(x)$.
$\mathcal{C} : \{0, 1\}^m \to \{0, 1\}^{m'}$	a boolean circuit
$\mathcal{C}_l : \{0, 1\}^m \to \{0, 1\}^{m'}$	layer l of the boolean circuit $\mathcal{C}$, consisting of m' boolean gates of fan-in at most two.

Table 1. Definitions of terms used in this work.

We also use the following conventions for clarity:

$i, j \in \{1, ..., d\}$	indices for neurons
$k, \ell, p \in \{1, ..., m\}$	indices for features
μ	amount of interference between near-orthogonal vectors
ε	error in the read-off of a boolean feature
s	A bound on the "sparsity"; we require $\lvert\lvert \boldsymbol{b}(x) \rvert\rvert_1 \leq s \ \forall \ x \in X$.

Table 2. Conventions used in this work.

We assume our terms satisfy the following asymptotic relationships in terms of the principal complexity parameter m (the number of features):

d is polynomial in m	so $d = \tilde{\Omega}(m^{\alpha+}), d = \tilde{O}(m^{\alpha-})$ for some finite exponents $0 < \alpha \leq \alpha_- < \infty$.
s is at worst polynomial in m,	so $s = O(m^{\beta})$. Note that this is different from the body, where we assumed s is a constant (so $\beta = 0$).
$s = \tilde{O}(d^{1/3})$.	This is a technical "sparsity" condition that will be useful for us.

Table 3. Asymptotic relationships between variables in this work.

B. Potential takeaways for practical mechanistic interpretability

Our motivation for studying these mathematical models is to glean insights about the computation implemented by real networks, that could have ramifications for the field of mechanistic interpretability, particularly the subfield focussed on taking features out of superposition in language models using sparse dictionary learning (Cunningham et al., 2023; Bricken et al., 2023; Tamkin et al., 2023; Bloom, 2024; Braun et al., 2024; Templeton et al., 2024). In order to render the models mathematically tractable, we have had to make idealising assumptions about the computation implemented by the networks.

1. Early work on superposition (Elhage et al., 2022) suggested that it may be possible to store exponentially many features in superposition in an activation space. On the other hand, early sparse dictionary learning efforts (Cunningham et al., 2023; Bricken et al., 2023; Bloom, 2024) learn dictionaries which are smaller than even the square of the dimension of the activation space. Our work suggests that the number of features that can be stored in superposition *and computed with* is likely to be around $\tilde{O}(d^2)$ (this is also the information-theoretic limit). We think that by using a dictionary size that scales quadratically in the size of the activations, while computationally challenging, this will likely lead to better performance on downstream tasks. We are heartened by more recent work by Templeton et al. (2024) which works with dictionaries that are closer to this size, and would encourage more systems-oriented work to scale to ever larger dictionaries.

2. The current mainstream sparse autoencoder (SAE) architecture used by Cunningham et al. (2023); Bricken et al. (2023); Bloom (2024); Templeton et al. (2024) and others uses ReLUs to read off feature values, in accordance with the toy model of superposition of Elhage et al. (2022) and features being ReLU-linearly represented. Our work suggests that networks may be more expressive when storing features ε-linearly. If so, this suggests that future work should consider sparse dictionary learning with alternative activation functions that only allow for removing errors of size ε, such as a *noise-filtering nonlinearity*

   $$\mathrm{NF}_\varepsilon(x) = \begin{cases} x & |x| > \varepsilon \\ 0 & |x| \leq \varepsilon \end{cases}.$$

   or nonlinearities that filter all but the k largest positive and largest negative preactivations. Notably, recent work by Rajamanoharan et al. (2024); Taggart (2024) finds suggestive evidence that the ProLU activation:

   $$\mathrm{ProLU}_\varepsilon(x) = \begin{cases} x & x > \varepsilon \\ 0 & x \leq \varepsilon \end{cases}$$

   outperforms the standard ReLU activation SAEs, which accords with the predictions in this work.

3. Previous work by Gurnee et al. (2023) found some features that were represented on a *small* set of neurons, even when they weren't represented on any singular particular neuron. In our constructions, feature representations end up distributed over a larger range of neurons. We expect that networks which employ superposition heavily to maximise their expressiveness are unlikely to have many sparse features that are localised to one or even a few neurons.

C. Additional discussion of various feature definitions

C.1. Formal statements and proofs for facts referenced in main body

We present formal statements and proofs that we referred to in Section 2.1. Note that without loss of generality, we can include the activation function a into our input set X, so we omit the use of a in this section.

Theorem 9 (Composition of linearly separable features)**.** *There exist a set of inputs X and two features f_1, f_2 weakly linearly represented in X such that there exists no MLP layer* MLP *such that either $f_1 \wedge f_2$ or $f_1 \vee f_2$ are linearly separable in* $\mathrm{MLP}(x)$.

Proof. (sketch) Let $X = [-1, 1]^2$ be the unit square in $\mathbb{R}^2$, and let $f_1(x) = \mathbf{1}(x_1 > 0)$ and $f_2(x) = \mathbf{1}(x_2 > 0)$ be the indicator functions of whether the first and second coordinates are greater than zero. There exists no MLP layer $\mathrm{MLP} : X \to \mathbb{R}^d$ of any width d such that $f_1 \wedge f_2$ is linearly separable in $\mathrm{MLP}(X)$.

To show this, it suffices to notice that any MLP layer has finite Lipschitz coefficient, and that any function weakly linearly representing $f_1 \wedge f_2$ or $f_1 \vee f_2$ will need to have arbitrarily high Lipschitz coefficient (since there exist points that are arbitrarily close to the separating hyperplanes of f_1 and f_2. □

Theorem 10 (Composition of ε-linearly represented features)**.** *For any set X and features f_1, f_2 that are ε-linearly represented in X, there exists a two neuron MLP $\mathrm{MLP} : X \to \mathbb{R}^2$ such that $f_1 \wedge f_2$ and $f_1 \vee f_2$ are ε'-linearly represented in $\mathrm{MLP}(X)$ for some ε'.*

Proof. (sketch) We use an MLP with two neurons MLP_1, MLP_2 with input weights equal to the read-off vectors of $\vec{r}_1, \vec{r}_2$. To read off $f_1 \wedge f_2$, we use the read-off vector $\vec{r}_{1\wedge 2}$ defined by $\vec{r}_{1\wedge 2}(x) = \mathrm{MLP}_1(x) + \mathrm{MLP}_1(x) - 3/4$. Similarly, to read off $f_1 \vee f_2$, we use the read-off vector $\vec{r}_{1\vee 2}(x) = \mathrm{MLP}_1(x) + \mathrm{MLP}_1(x) - 1/4$. □

In fact, by allowing for wider MLPs, it is fairly easy to construct an MLP $\mathrm{MLP} : X \to \mathbb{R}^d$ such that $f_1 \wedge f_2$ and $f_1 \vee f_2$ are also ε-linearly represented in $\mathrm{MLP}(X)$ (that is, with equal error). We leave the construction of this MLP as an exercise for the reader.

D. Additional definitions and formalism

Here we provide additional definitions required for our proofs in Appendix E.

D.1. Negligible probabilities

Most results in this paper are proven *outside negligible probability*. This is a standard notion in complexity theory and cryptography (Bellare, 2002), with the following formal definition:

Definition 5. *Let $\{E_n\}_{n=1}^{\infty}$ be a sequence of events parameterized by n. We say that E_n is true* ***with negligible probability*** *(w. n. p.) if for any polynomial exponent $c \in \mathbb{N}$, there exists some constant $N_c \in \mathbb{N}$ such that $P(E_n) < O(n^{-c})$ for all $n > N_c$. Similarly, we say that E_n is true* ***outside negligible probability*** *(o. n. p.) if its complement $\overline{E_n}$ is true with negligible probability.*

Intuitively, the reason why this probability is "negligible" is that the union of polynomially many events of negligible probability also has negligible probability. As we never consider more networks requiring more than polynomially many operations, we can ignore events of negligible probability at each step when performing asymptotic analysis, which greatly simplifies our proofs.

Example 1. Let $\boldsymbol{b}$ be a random boolean vectors of length n. Then outside negligible probability, $\boldsymbol{b}$ has between $n/2 + \log(n)\sqrt{n}$ and $n/2 - \log(n)\sqrt{n}$ zeroes.

This follows from the central limit theorem. (Note that if we used $\sqrt{\log(n)}\sqrt{n}$, the result would be false!)

D.2. Precise and mixed emulations

The parameters in the models $\mathcal{M}_w$ in the proofs of our emulation results depend on random matrices of ±1's and 0's, hence can be understood as suitable random variables. In terms of this point of view, we make the following definition.

Suppose that $\mathcal{C} : \{0,1\}^m \to \{0,1\}^{m'}$ is a boolean circuit with input size m. We always assume that the output size m' and the depth are at most polynomial in m. Let $\mathcal{B} \subset \{0,1\}^m$ be a class of inputs (usually characterized by a suitable sparsity property). Let $\varepsilon < 1$ be an interference parameter.

Definition 6. *An* ε-precise emulation *of $\mathcal{C}$ (on input class $\mathcal{B}$) is a triple of data $(\Phi, \mathcal{M}_w, \mathbf{R})$ all possibly depending on random parameters where $\Phi \in \mathrm{Mat}_{d_{\mathrm{in}} \times m}$ is a feature matrix, $\mathbf{R} \in \mathrm{Mat}_{m' \times d_{\mathrm{out}}}$ is a readoff matrix and*

$$\mathcal{M}_w : \mathbb{R}^{din} \to \mathbb{R}^{dout}$$

is a (not necessarily linear) function given by a neural net, with the following property:

For any $\boldsymbol{b} \in \mathcal{B}$, we have, outside negligible probability,

$$||\mathbf{R} \circ \mathcal{M}_w \circ \Phi(\boldsymbol{b}) - \mathcal{C}(\boldsymbol{b})||_{\infty} < \varepsilon.$$

Importantly, we do not consider the boolean circuit $\mathcal{C}$ or the input $\boldsymbol{b} \in \mathcal{B}$ to be random variables, and the randomness involved in the negligible probability statement is purely in terms of the parameters that go into the emulation scheme

$(\Phi, \mathcal{M}_w, \mathbf{R})$. In particular, this guarantees that if the boolean input $\boldsymbol{b}$ is generated in a non-random way (e.g., adversarially), an emulation nevertheless guarantees (in the "negligible probability sense") safe performance on b so long as the parameters of the emulation were chosen randomly.

It will be useful to extend the notion of emulation to one which correctly approximates $\mathcal{C}$ on inputs $x \in \mathbb{R}^{d\text{in}}$ which represent a boolean input $\boldsymbol{b} \in \{0,1\}^m$ not in the sense of "pure superposition" $x = \Phi(\boldsymbol{b})$ but in the sense of "read-off",

$$||\mathbf{R}_{\text{in}}(x) - \boldsymbol{b}||_\infty < \varepsilon_{\text{in}}.$$

Here $\mathbf{R}_{\text{in}} \in \text{Mat}_{\times m} d_{\text{in}}$ is a readoff matrix that should be thought of as a noisy inverse to the feature matrix on sparse inputs. Formally, we make the following definition. Here we will assume that the matrix $\mathbf{R}_{\text{in}}$ was generated at an earlier stage of the computation, and does not depend on random variables.

Fix a circuit $\mathcal{C} : \{0,1\}^m \to \{0,1\}^{m'}$, a class of inputs $\mathcal{B} \subset \{0,1\}^m$, and an "input readoff" matrix $\mathbf{R}_{\text{in}} \in \text{Mat}_{m \times d_{\text{in}}}$. Let $\varepsilon_{\text{in}}, \varepsilon_{\text{out}}$ be two interference parameters.

Definition 7. *A* mixed emulation of $\mathcal{C}$ with precision $\varepsilon_{\text{in}} \to \varepsilon_{\text{out}}$ *(on input class $\mathcal{B}$ and relative to a fixed input readoff matrix $\mathbf{R}_{\text{in}}$) is a pair of data $(\mathcal{M}_w, \mathbf{R}_{\text{out}})$ both possibly depending on random parameters where $\mathbf{R} \in \text{Mat}_{m' \times d_{\text{out}}}$ is a readoff matrix and*

$$\mathcal{M}_w : \mathbb{R}^{d\text{in}} \to \mathbb{R}^{d\text{out}}$$

is a (not necessarily linear) function given by a neural net, with the following property:

For any boolean input $\boldsymbol{b} \in \mathcal{B}$ and $x \in \mathbb{R}^{d\text{in}}$ satisfying

$$||\mathbf{R}_{\text{in}}(x) - \boldsymbol{b}||_\infty < \varepsilon_{\text{in}},$$

we have, outside negligible probability,

$$||\mathbf{R} \circ \mathcal{M}_w(x) - \mathcal{C}(\boldsymbol{b})||_\infty < \varepsilon_{\text{out}}.$$

Remark 11. Note that if it is impossible to accurately represent $\boldsymbol{b}$ via the matrix $\mathbf{R}$, i.e., to satisfy $||\mathbf{R}(x) - \boldsymbol{b}||_\infty < \varepsilon_{\text{in}}$, then the notion of mixed emulation is vacuous (any neural net would satisfy it for tautological reasons). We will generally apply this notion in contexts where such representations are possible (for example, with via a suitable feature matrix $x = \Phi(\boldsymbol{b})$).

Here as before we do not consider the boolean circuit $\mathcal{C}$ or the input $\boldsymbol{b} \in \mathcal{B}$ to be random variables, and in addition the representation x and the input readoff matrix $\mathbf{R}_{\text{in}}$ are assumed fixed. So the randomness involved in the negligible probability statement is purely in terms of the parameters that go into the pair $(\mathcal{M}_w, \mathbf{R})$.

E. Precise statements and proofs of theorems

Let m be a parameter associated to the length of a boolean input. For the remainder of this section, we will work with real parameters $\alpha, \beta_{\text{in}}, \beta_{\text{out}}, \gamma$ which do not scale with m and corresponding to scaling exponents. We impose the following asymptotic relationships on parameters m (length of boolean input), $d = d_{\text{in}}$ (width of emulating neural net), s (sparsity, i.e., number of 1 values, of suitable boolean variables), ε_{in} (incoming interference, if applicable) and ε_{out} (outgoing interference):

$$m = \tilde{\Omega}(r^\alpha) \quad (1)$$

$$\varepsilon_{\text{in}} = \tilde{\Omega}(r^{-\beta\text{in}}) \quad (2)$$

$$\varepsilon_{\text{out}} = \tilde{O}(r^{-\beta\text{out}}) \quad (3)$$

$$s = \tilde{O}(r^\gamma). \quad (4)$$

More precisely, we assume that a large parameter m is given and the $O(\text{polylog}(m))$ scaling factors implicit in the $\tilde{O}, \tilde{\Omega}$ asymptotics can be chosen in a suitable way to make the results hold.

E.1. Emulation of AND layer

In this section we prove a generalization of Theorem 3.2.

Let $\Gamma \subset \{1, \ldots, m\}^{[2]}$ be the edges of a graph (here the superscript [2] denotes the "exterior power" of a set, i.e., the set of $\binom{m}{2}$ unordered pairs). Assume that the number of edges $|E_\Gamma| = \tilde{O}(m)$. Let $\mathcal{C}_\Gamma : \{0,1\}^m \to \{0,1\}^{E_\Gamma}$ be the circuit with value

$$\mathcal{C}_\Gamma(\boldsymbol{b})_{(k,\ell)} = \boldsymbol{b}_k \wedge \boldsymbol{b}$$

at the unordered pair $(k, \ell) \in E_\Gamma$ corresponding to an edge of Γ. We think of $\mathcal{C}_\Gamma$ as the (not quite universal) circuit that takes AND's of pairs of features in Γ and returns a boolean vector of roughly the same size.

We will show that this circuit can be emulated with suitably small interference on the output.

The proof is very similar to the proof of the error correction theorem above (Theorem **??**), in particular with the main argument controlled by a subset $\Sigma \subset \{1, \ldots, m\} \times \{1, \ldots, d\}$, with m the number of *edges* of Γ (i.e., outputs of the circuit).

There are however two main differences.

1. What we read from each subset $\Sigma_{k,\ell}$ associated to an edge $(k, \ell) \in \Gamma$ is a the result of a nonlinearity applied to a *sum* of two random ± 1 vectors ϕ_k, ϕ_ℓ (associated to the two inputs k, ℓ), that returns (up to small error) the sum of neurons in of Σ_{ij} where the signs of ϕ_k and ϕ_ℓ are both 1.

2. To control interference issues, we need to carefully partition the graph Γ into pieces with a certain asymptotic "balanced" property (see Lemma **??**).

3. The output interference is $\tilde{O}(\sqrt{\frac{s^2}{d}}$ instead of $\tilde{O}(\sqrt{\frac{s}{d}}$ since there are $O(s^2)$ active output features (corresponding to pairs of features that are on).

Theorem 12 (Targeted superpositional AND)**.** *Let m be an integer and $\Gamma \subset \{1, \ldots, m\} \times \{1, \ldots, m\}$ a graph. Assume we have a readoff matrix $\mathbf{R}_{\text{in}} \in \text{Mat}_{m \times d}$ that maps a d-dimensional space to an m-dimensional space, and let $s = o(\sqrt{m})$ be a sparsity parameter (either polynomial or polylogarithmic in m). Let ε_{in} be an interference parameter.*

Assume that we have $\varepsilon_{\text{in}}^2 md\sqrt{d/s} = \tilde{O}(1)$ is bounded by some sufficiently small inverse polylogarithmic expression in m. Then there exists a single-layer mixed emulation $\mathcal{M}_w(x) = \text{ReLU}(W_{in}x + w_{bias})$ of the universal AND circuit $\mathcal{C}_{\text{uand}}$ (together with an "output readoff" matrix $\mathbf{R}_{\text{out}}$) such that $\mathcal{M}_w$ is an emulation of $\mathcal{C}_\Gamma$ on the input class $\mathcal{B} = \mathcal{B}_s$ of boolean vectors of sparsity $\leq s$, with precision $\varepsilon_{\text{in}} \to \varepsilon_{\text{out}}$, for $\varepsilon_{\text{out}} = \tilde{O}\left(\sqrt{\frac{s^2}{d}}\right)$.

Before proving the theorem, we note that our UAND statements are corollaries:

Corollary 13 (U-AND with basis-aligned inputs)**.** *Fix a sparsity parameter $s \in \mathbb{N}$. Then for large input length m, there exists a single-layer neural network $\mathcal{M}_w(x) = \text{MLP}(x) = \text{ReLU}(W_{in}x + w_{bias})$ that ε-linearly represents the universal AND circuit $\mathcal{C}_{\text{UAND}}$ on s-sparse inputs, with width $d = \tilde{O}_m(1/\varepsilon^2)$ (i.e. polylogarithmic in m).*

This follows from the fact that the incoming interference $\varepsilon_{\text{in}} = 0$ since the incoming feature basis is basis-aligned.

Corollary 14 (U-AND with inputs in superposition)**.** *Let $s \in \mathbb{N}$ be a fixed sparsity limit and $\varepsilon < 1$ a fixed interference parameter. There exists a feature encoding Φ and single-layer neural net $\mathcal{M}_w(x) = \text{MLP}(x) = \text{ReLU}(W_{in}x + w_{bias})$ with input size m_{in} and width $d = \tilde{O}(\sqrt{m_{\text{in}}}/\varepsilon^2)$, such that $\mathcal{M}_w \circ \Phi$ ε-linearly represents $\mathcal{C}_{\text{UAND}}$ on all s-sparse inputs $\boldsymbol{b}$.*

This follows by restricting all but $m_{\text{in}} = \sqrt{m}$ input features to 0 and taking Γ to be the complete graph on vertices $\{0, \ldots, m_{\text{in}}\}$.

Now we prove the theorem.

Proof. We begin by considering a simpler case. We say that a graph Γ with m edges is *self-balanced* if each vertex has degree at most $\tilde{O}(1)$ (some fixed polylogarithmic-in-m bound).

Suppose Γ is self-balanced. Define $A := \sqrt{d/s}$. For each edge $(k, \ell) \in \Gamma$, choose at random a subset $\Sigma_{k\ell} \subset \{1, \ldots, d\}$ of size within a polylog error of A. Write also

$$\Sigma_k = \bigcup_{\ell | (k,\ell) \in \Gamma} \Sigma_{k,\ell}.$$

Write down feature vectors $\vec{\phi}_{k\ell} = \sum_{i \in \Sigma_k} \pm \vec{e}_i$, with signs $\sigma_{k,i}$ chosen independently and randomly for each k, i. For a pair $k, \ell \in \Gamma$, define the vector $\vec{r}_{k,\ell}$ to be the indicator of the set of neurons

$$\Sigma_{k,\ell}^{\text{out}} := \{i \in \{1, \ldots, d\} \mid \sigma_{k,i} = \sigma_{\ell,i} = 1\},$$

Note that $|\Sigma_{k,\ell}^{\text{out}}|$ has, o. n. p., within a polylog difference from $\frac{1}{4}|\Sigma_{k,\ell}| = \frac{A}{4}$ elements.

Write

$$\vec{\phi}_k^{\text{in}} := \sum_{\ell | (k,\ell) \in \Gamma} \vec{\phi}_{k,\ell},$$

Note that this is a indicator function of a union polylog-many independently chosen sets of size A. Write Φ^{in} for the $m \times d$ matrix with columns $\vec{\phi}_k^{\text{in}}$.

Now we define the emulation net to be

$$\mathcal{M}_{w\Gamma}(x) = \frac{4}{A}\text{ReLU}(\Phi^{\text{in}}(x) - 1).$$

We note that (outside interference and collision errors of frequency bounded o. n. p. by $\tilde{O}(\varepsilon_{\text{out}})$,) we have

$$\text{ReLU}(\Phi^T(\boldsymbol{b})) - 1)_i = \begin{cases} 1, & \exists k, \ell \in S \text{ with } i \in \Sigma_{k,\ell} \text{ and } \sigma_{k,i} = \sigma_{\ell,i} = 1 \\ 0, & \text{otherwise,} \end{cases}.$$

Here as before we take $S \subset \{1, \ldots, m\}$ for the set of features that are on.

Analogously to our proof of Lemma 24's part **??** we see that the difference $\Phi^T(\boldsymbol{b}) - \Phi^T(\mathbf{R}_{\text{in}}(x))$ is (o. n. p.) bounded by $o(1)$, and thus we are done just as in the previous lemma.

For general graphs Γ, we might have an issue if some vertices have very high degree; if one were to try to run the same proof, their corresponding features would then admit unmanageably high interference.

To fix this, we note that in order to emulate $\mathcal{C}_\Gamma$ it is sufficient (up to polylogarithmically increasing the number of neurons) to emulate $\mathcal{C}_{\Gamma_1}, \ldots, \mathcal{C}_{\Gamma_T}$ for some polylogarithmic collection of graphs Γ_t with $\cup_t \Gamma_t = \Gamma$. We now split an arbitrary graph Γ into subgraphs with a nice "balanced" property.

Let $a, b \in \mathbb{R}$ be parameters. We say that a graph is a, b-balanced if it is bipartite on a pair of disjoint subsets of vertices $V_0, V_1 \subset \{0, \ldots, m\}$, such that $|V_0| = a$, $|V_1| = b$ and each vertex in V_0 has degree at most m/a and each vertex in V_1 has degree at most m/b. We say a graph $\Gamma \subset \{0, \ldots, m\}^{[2]}$ is balanced if it is a, b-balanced for some a, b.

It can be shown using an inductive argument that any graph Γ with m edges can be written as a union of $\text{polylog}(m.)$

Now it remains to show that the theorem holds for a balanced graph. Indeed, suppose that Γ has vertices supported on $V_0 \sqcup V_1 \subset \{1, \ldots, m\}$ and is a, b-balanced. Suppose (WLOG) that $a \leq b$. Then we randomly partition the neurons $\{1, \ldots, d\}$ into a roughly equal sets Σ_k for $k \in V_0$ (equivalently, we choose a random map $\{1, \ldots, d\} \to V_0$ and define Σ_k to be the preimage of k). We then choose for $\ell \in V_1$ the set $\Sigma_{k,\ell}$ to be a random subset of size about $\sqrt{d/s^2}$ inside Σ_k, and define $\Sigma_\ell = \cup_{k|(k,\ell) \in \Gamma}$. We finish the argument by bounding the errors in the same way as in the self-balanced case, concluding the proof. □

E.2. Universal AND with inputs in superposition

We use the conventions from Section A. We make an additional assumption, that our inputs $\vec{a}^{(0)}(x)$ for $x \in X$ approximately lie on a sphere of suitable radius. Note that if $m = d$ and the feature basis $\vec{\phi}_i$ is an orthonormal basis, then $|\Phi(\boldsymbol{b})| = \sqrt{||\boldsymbol{b}||_1}$, so the ℓ_2 norm of the embedding is the square root of the sparsity. If the sparsity $||\boldsymbol{b}||_1$ is *exactly* s and the feature interference parameter μ is sufficiently small compared to the sparsity bound s, we still have $|\Phi(\boldsymbol{b})| \approx \sqrt{s}$ (with some suitable bound — in general, it will be $\tilde{O}(\mu s^{1.5})$). If instead, we assume only that the boolean features $f_i(\boldsymbol{b})$ are ε-linearly represented for suitable $\varepsilon > \frac{1}{\sqrt{d}}$, in general we cannot guarantee that $|\vec{a}^{(0)}(x)| \approx \sqrt{s}$; rather, we will have $|\vec{a}^{(0)}(x)| = \tilde{\Omega}(\sqrt{s})$ since especially for small s, the norm might be significantly increased by adding a large vector that is almost-orthogonal to all features (and thus doesn't affect the linear representability of the f_i). This observation allows us, in principle, to write down a vector with some suitable norm in $\tilde{\theta}(\sqrt{s})$ which ε-linearly represents a very sparse boolean vector $\boldsymbol{b}$ with $||\boldsymbol{b}||_1 << s$. We show how to modify inputs with unknown bounded sparsity $||\boldsymbol{b}(x)||_1 < s$ to have an (approximately) constant norm in the following section. For now, we assume in addition to $||\boldsymbol{b}(x)||_1 < s$ that all our inputs have norm equal to some $s_0 = \tilde{O}(\sqrt{s})$ up to a small error.

Theorem 15. *Let $m, d, X, \Phi, \varepsilon = \varepsilon_0, \mu, s$ be as in Appendix A. Let r be a parameter so that $r^2 = \tilde{O}(s)$. Assume in addition to the conditions on X, Φ in Appendix A that for any input $x \in X$, we have*

$$|\vec{a}^{(0)}(x)| = r + \tilde{O}(\frac{\sqrt{s}}{\sqrt{d}}),$$

i.e., the inputs lie approximately on a sphere of radius r.

Let $W \in \mathrm{Mat}_{d\times d}$ be a random weight matrix with i.i.d. Gaussian-distributed entries, and let $\vec{a}^{(1)}(x) = \mathcal{M}_w(\vec{a}^{(0)}(x)) := \mathrm{ReLU}(Wx)$ be the associated neural net. Then there exist some

$$\varepsilon^{(1)} = \tilde{O}\big(\max(s\mu, \sqrt{s}\varepsilon, \sqrt{s/d})\big)$$

and

$$\mu^{(1)} = \tilde{O}\big(\max(\sqrt{1/d}, \mu)\big),$$

such that the boolean function $f_{k\wedge\ell}(x) := f_k(x) \wedge f_\ell(x)$ is $\varepsilon^{(1)}$-linearly represented by a feature vector $\vec{\phi}^{(1)}_{k\wedge\ell} \in \mathbb{R}^d$, outside negligible probability (in the entries of W). Moreover, up to rescaling by a fixed scalar, the feature vectors $\vec{\phi}_{k\wedge\ell}$ form an almost-orthogonal collection with feature interference parameter $\mu^{(1)}$.

Corollary 16. *The result of Theorem 15 is true with the assumption $|\vec{a}^{(0)}(x)|^2 = r^2 + \tilde{O}(\varepsilon s)$ (that inputs are close to a sphere) replaced by $|\vec{a}^{(0)}(x)|^2 = \tilde{O}(s)$, at the cost of increasing the depth of the neural network $\mathcal{M}_w$ from 1 to 3.*

Proof. (Of corollary.) This follows by chaining the neural network constructed in this theorem with the "norm-balancer network" constructed in Appendix E.3 (independent from this one). □

The idea of the proof of Theorem 15 is derived from the quadratic activations case, $\mathcal{M}_w(\vec{x}) = Q(W\vec{x})$, where Q is the function that squares entries of a vector coordinatewise. Let $a^i_k = W(\vec{\phi}_k)^i$ (for $i \in \{0, \ldots, d-1\}$) be the coordinates of the preactivation vector $W(\vec{\phi}_k)$ associated to the kth boolean bit.

One can show using the theory of quadratic forms that the readoff vector $R^i_{k,\ell} = a^i_k a^i_\ell$ gives a valid readoff direction to show ε-strong linear separation of the boolean expression $\boldsymbol{b}_k \wedge \boldsymbol{b}_\ell$ (o. n. p.). We will show that a similar strategy works for an arbitrary (reasonable, and in particular nonlinear) activation function, including ReLU.

Write down the unnormalized model $\mathcal{M}_w{}^u(\vec{x}) := \mathrm{ReLU}(W(\vec{x}))$. Define $\vec{\phi}'_k = W\vec{\phi}_k$ to be the preactivation under this model of $\vec{\phi}_k$. Define the unnormalized readoff matrix for the UAND coordinate associated to the pair of features k, ℓ as follows:

$$\vec{r}^{\,i}_{k,\ell} = ((\vec{\phi}'_k)_i \cdot (\vec{\phi}'_\ell)_i),$$

where $\mathrm{sign}(x)$ is the sign function that returns $-1, 0, 1$ depending on whether x is negative, 0 or positive, respectively.

Remark 17. Note that as we care about the existence of a linear representation rather than a learnable formula for it, the readoff doesn't have to depend continuously on the parameters. However having continuous dependence is also possible; in particular, it would also be reasonable to make the dependence continuous; indeed, the readoff vector with coordinates $a^i_k \cdot a^i_\ell$ (same as for quadratic activations) would also work, with an alternative normalization; the important property of the readoff function is that it is odd in each of the x and y coordinates independently, and that it does not have wild asymptotic behavior. We use the discrete "sign" function for the readoff for convenience.

The crucial observation is the following simple lemma. For a given input x, let $\vec{a}(x)$ be the corresponding embedding. Let

$$\vec{a}(x)^\Lambda := \vec{a}(x) - f_k(x)\vec{\phi}_k - f_\ell(x)]\vec{\phi}_\ell$$

(the "hat" notation denotes that we are "skipping" information about features k and ℓ in the embedded input $\vec{a}(x)$; it linearly represents the modification of the boolean vector $\boldsymbol{b}(x)$ that zeroes out the kth and ℓth coordinates).

Lemma 18. *Suppose Φ, k, ℓ, and $\boldsymbol{b}$ are fixed. Then in the context of the theorem above, the unnormalized readoff $\mathbf{R}^u_{k,\ell}(\mathcal{M}_w(\Phi(\boldsymbol{b})))$ is a sum of d i.i.d. variables of the form $F(x_i, y_i, z_i)$, where $F(x,y,z) = (x)(y)\mathrm{ReLU}(\boldsymbol{b}_k(x)x + \boldsymbol{b}_\ell(x)y + z)$ and the triple (x_i, y_i, z_i) is drawn from the distribution $(0, \Sigma)$ where*

$$\Sigma = \begin{pmatrix} ||\vec{\phi}_k||_2^2 & \vec{\phi}_k \cdot \vec{\phi}_\ell & \vec{\phi}_k \cdot \vec{x}^\Lambda \\ \vec{\phi}_k \cdot \vec{\phi}_\ell & ||\vec{\phi}_\ell||_2^2 & \vec{\phi}_\ell \cdot \vec{x}^\Lambda \\ \vec{\phi}_k \cdot \vec{x}^\Lambda & \vec{\phi}_\ell \cdot \vec{x}^\Lambda & ||\vec{x}^\Lambda||^2 \end{pmatrix}.$$

Proof. Write $x_i = (\vec{\phi}'_k)_i, y_i = (\vec{\phi}'_\ell)_i, z_i = W\vec{a}(x)^\Lambda$ be the neuronal coordinates of the corresponding activations. Then $(R^u_{k,\ell})_i = (x_i)(y_i)$ and

$$\mathcal{M}_w{}^u(\vec{a}(x))_i = \mathrm{ReLU}\big(W(\vec{a}(x))_i\big) = \mathrm{ReLU}(\boldsymbol{b}(x)_k x_i + \boldsymbol{b}(y)_k y_i + z_i).$$

It remains to show that (x_i, y_i, z_i) are drawn according to the Gaussian distribution $(0, \Sigma)$. This follows from the standard result that applying a Gaussian-distributed matrix with entries in $(0, 1/d)$ to a collection of vectors $\vec{v}_1, \ldots, \vec{v}_n$ is distributed as a (possibly singular) Gaussian with PSD covariance matrix $\Sigma_{k\ell} = \vec{v}_k \cdot \vec{v}_\ell$. □

Now our interference bounds imply that the triple $(x_i, y_i, z_i + \boldsymbol{b}_k x_i + \boldsymbol{b}_\ell y_i)$ are distributed according to a matrix of the form

$$\begin{pmatrix} 1 + O(\mu) & O(\mu) & \boldsymbol{b}_k + O(\varepsilon) \\ O(\mu) & 1 + O(\mu) & \boldsymbol{b}_\ell + O(\varepsilon) \\ \boldsymbol{b}_k + O(\varepsilon) & \boldsymbol{b}_\ell + O(\varepsilon) & r^2 + \tilde{O}(s/\sqrt{d}). \end{pmatrix}$$

Let $s' := r^2 - \boldsymbol{b}_k - \boldsymbol{b}_\ell$ and $r' := \sqrt{s'}$.

Now o.n.p., we can assume that $x_i, y_i \in \tilde{O}(1)$ and $z_i \in \tilde{O}(r)$. Since F grows linearly, we see that $F(x_i, y_i, z_i) \in \tilde{O}(r)$ o.n.p. We can now apply Bernstein's inequality 32 to get that, o.n.p.,

$$\sum_{i=1}^{d} F(x_i, y_i, z_i) = d[\mathbb{E}_{(x,y,z)\sim(0,\Sigma)} f(x,y,z) + \tilde{O}(r/\sqrt{d})].$$

Now since $r = \tilde{O}(\sqrt{s})$ and $|(r')^2 - r^2|$ is an integer equal to at most 2 (the sum of two feature readoffs of $\vec{a}$), the error term in the Bernstein inequality is bounded by $\tilde{O}(r'/\sqrt{d})$. It remains to estimate the expectation

$$E := \mathbb{E}_{(x,y,z)\sim(0,\Sigma)} F(x, y, \bar{z}).$$

Assume that $\boldsymbol{b}(x)$ has nonzero coordinates other than at k, ℓ, so that $r' = \Omega(1)$ (the case where $\boldsymbol{b}(x)$ only has nonzero coordinates on a subset of $\{k, \ell\}$ can be handled similarly and more easily). In this case, we add a new notation

$$F'(x, y, z') := F(x, y, s'z') = (x)(y)\mathrm{ReLU}(r'\bar{z} + \boldsymbol{b}_k x + \boldsymbol{b}_\ell y),$$

where the third input of F is rescaled to make the distribution on (x, y, z') closer to the identity Gaussian. Let Σ' be the distribution on (x, y, z'), given by

$$\Sigma' = \mathrm{diag}(1, 1, (r')^{-1})\Sigma\mathrm{diag}(1, 1, (r')^{-1}).$$

Since the two differ by a reparametrization, the expectation of F' on $(0, \Sigma')$ is equal to the expectation of F on $(0, \Sigma)$.

Let $X' = (0, \Sigma')$ and $X'_0 = (0, \Gamma)$, both on $\mathbb{R}^3$. Our various interference bounds imply that the difference $\Sigma - \Gamma$ is bounded by

$$\delta := \tilde{O}\big(\max(\frac{\sqrt{s}}{\sqrt{d}}, \frac{\varepsilon}{\sqrt{s}}, \mu)\big).$$

This means that the total variational difference between X and X' is bounded by $O(\delta)$. Now the expectation F' on X, X_0 are not affected, up to negligible terms, by (x, y, z) outside some constant $\tilde{O}(1)$, and here F' is bounded by $\tilde{O}(r)$. Thus we have

$$|\mathbb{E}_{(x,y,z')\sim X} F'(x, y, z') - \mathbb{E}_{(x,y,z')\sim X_0} F'(x, y, z')| = \tilde{O}(r\delta).$$

It remains to estimate the mean

$$E_0 := \mathbb{E}_{(x,y,z')\sim X_0'} F'(x,y,z') = \mathbb{E}_{(x,y,z)\sim X_0} F(x,y,z),$$

where $X_0 = (0, \mathrm{diag}(1, 1, (d')^2))$.

Up to symmetry, we have three cases depending on the k and ℓ coordinates of $\boldsymbol{b} = \boldsymbol{b}(x)$ associated to our input:

- $\boldsymbol{b}_k = \boldsymbol{b}_\ell = 0$,
- $\boldsymbol{b}_k = 0, \boldsymbol{b}_\ell = 1$,
- $\boldsymbol{b}_k = \boldsymbol{b}_\ell = 1$.

The expectation calculation in the first two cases are trivial: if $\boldsymbol{b}_k$, is zero, then each F is odd in the x, resp., y coordinate, so since the distribution X_0 is independent Gaussian, the mean is

$$E_0 = 0.$$

It remains to consider the case $\boldsymbol{b}_k = \boldsymbol{b}_\ell = 1$, i.e., the “interesting” case where $\wedge(\boldsymbol{b}_k, \boldsymbol{b}_\ell) = 1$. We write down the integral expression

$$E_0 := \mathbb{E}_{(x,y,z)\sim X_0} Q_i(x,y,z) = \int (x)(y) \mathrm{ReLU}(x+y+z) p_0(x,y,z) dxdydz, \tag{5}$$

for $p_0(x,y,z)$ the pdf of $X_0 = (0, \mathrm{diag}(1,1,s'))$. We would like to show this value is positive and bound it from below (to show eventually that the mean in the CLT dominates the errors). We use x,y-symmetry to rewrite the integral as

$$A = 2\int_{x\le y} (x)(y)\mathrm{ReLU}(x+y+z)p_0(x,y,z).$$

Since the independent Gaussian $p_0(x,y,z)$ is symmetric in the x and y coordinates, we can collect $\pm x, \pm y$ terms together to write

$$E_0 = 2\int_{0\le x\le y} p(x,y,z)\big(\mathrm{ReLU}(x+y+z) - \mathrm{ReLU}(x-y+z) - \mathrm{ReLU}(-x+y+z) + \mathrm{ReLU}(x+y+z)\big).$$

We split the domain up further into five terms,

$$E_0 = A^{--} + A^{-} + A^{0} + A^{+}A^{++},$$

into regions on which the relus are constantly 0 or nonnegative linear functions:

$$\begin{aligned}
A^{--} \quad & =2\int_{0\le x\le y, z\le -x-y} p_0(x,y,z)dxdydz \cdot 0 \\
A^{-} \quad & =2\int_{0\le x\le y, -x-y\le z\le x-y} p_0(x,y,z)(x+y+z) \\
A^{0} \quad & =2\int_{0\le x\le y, x-y\le z\le y-x} p_0(x,y,z)dxdydz \left((x+y+z) - (-x+y+z)\right) \\
& =2\int_{\dots} p_0(x,y,z)dxdydz \, (2x) \\
A^{+} \quad & =2\int_{0\le x\le y, y-x\le z\le x+y} p_0(x,y,z)dxdydz \, (x+y+z) - (-x+y+z) - (x-y+z) \\
& =2\int_{\dots} p_0(x,y,z)dxdydz \left(x+y-z\right) \\
A^{++} \quad & =2\int_{0\le x\le y, z\ge x+y} p_0(x,y,z)dxdydz \left((x+y+z) - (-x+y+z) - (x-y+z) + (-x-y+z)\right) \\
& =0.
\end{aligned}$$

Note in particular that each term above is nonnegative on its domain (for A^+, this is because the domain includes the inequality $z \leq x + y$). Thus in particular, $E \geq A^0$. Since the integrand is positive, we can get a lower bound by restricting the domain:

$$A^0 \geq 2 \int_{x \leq 1, y \geq 2, -1 \leq z \leq 1} 2p_0(x, y, z) dxdydz,$$

using that the integrand is $2x \geq 2$. This is, equivalently, twice the probability that $|x| \geq 1, |y| \geq 2, |z| \leq 1$, for (x, y, z) drawn from $p_0(x, y, z) = \sigma_{0,1}(x)\sigma_{0,1}(y)\sigma_{0,r^2-2}(z)$. By independence of p_0, this is a product of 3 terms. The probability distributions on x, y are fixed unit Gaussians, so the corresponding terms are $O(1)$, and so the mean has (up to an $O(1)$ constant) the same asymptotic as the third term, which is

$$P_{z \sim \sigma_{0,r^2-2}}(|z| < 1) = O(1/r) = \tilde{\Theta}(1/\sqrt{s}).$$

The Bernstein bound applied to d i.i.d. such variables now gives us o.n.p.

$$\sum_{i=1}^{d} F(x_i, y_i, z_i)_{(x_i, y_i, z_i) \sim X_0} = d \cdot E_0 + \sqrt{d}\tilde{O}(r).$$

Incorporating error terms, we get

$$\vec{r}^{u}_{k,\ell}(\mathcal{M}_w{}^u(\vec{a}(x))) = d \cdot E_0 + \sqrt{d}\tilde{O}(r) + d\tilde{O}(r\delta).$$

We now normalize:

$$\mathcal{M}_w(\vec{a}) := \frac{\mathcal{M}_w{}^u(\vec{a})}{\sqrt{d}E_0} \tag{6}$$

$$\vec{r}_{k,\ell} := \frac{\vec{r}^{u}_{k,\ell}}{\sqrt{d}}. \tag{7}$$

Then if $f_k(x) \wedge f_\ell(x) = 1$, then (o.n.p.)

$$\vec{r}_{k,\ell}(x) = 1 + \frac{\tilde{O}(r)}{\sqrt{d}} + \tilde{O}r\delta.$$

Alternatively if $f_k(x) \wedge f_\ell(x) = 0$, the expectation is zero and we are left with the error term,

$$\vec{r}_{k,\ell}(x) = \frac{\tilde{O}(r)}{\sqrt{d}} + \tilde{O}r\delta$$

The theorem follows. □

E.3. Norm-balancer network

In this section, we prove a technical result that was needed in the previous section. Namely, at one point we assumed that the norm of our inputs $\vec{a}_0(x)$ are (o.n.p., and up to a multiplicative error of $1 + \tilde{O}(\frac{1}{\sqrt{d}})$) equal to a specific value λ, which is related to the sparsity by a bound of the form $\lambda = \tilde{O}(\sqrt{s})$. It is not difficult to guarantee this if we know the exact sparsity of the sparse boolean vector $s_{exact} = ||\boldsymbol{b}_0||_1$. However, in the process of chaining together multiple boolean circuits, we would like to allow the exact sparsity of intermediate layers to vary (so long as it is bounded by s), even if the exact sparsity of the input layer is fixed. In this section we give a two-layer neural network mechanism that allows us to circumvent this issue by modifying all inputs $\vec{a}_0(x)$ to have roughly the same norm, equal to some specific value $\sqrt{s_0} = \tilde{O}\sqrt{s}$.

We note that while it seems plausible that real neural networks share properties in common with the past two artificial neural nets we constructed (error correction and universal AND), the neural net constructed here

Theorem 19. *Let $s_0 = \tilde{O}(\sqrt{d})$ be a sparsity parameter. There exists a 2-layer neural net* $\mathrm{balance}_{s_0} : \mathbb{R}^d \to \mathbb{R}^d$ *depending on random parameters, with hidden layers of width $O(d)$, with the following property.*

Suppose that $\vec{\phi}_1, \ldots, \vec{\phi}_d$ is a collection of features of length < 2, and $\vec{a}_x$ is an input satisfying $|\vec{a}_x| < \sqrt{s_0}$. Then

1. $|\mathrm{balance}(\vec{a}_x)| = \sqrt{s_0} \cdot (1 + \tilde{O}(1/\sqrt{d}))$

2. $\vec{a}_x \cdot \vec{\phi}_k - \text{balance}(\vec{a}_x) \cdot \vec{\phi}_k = \tilde{O}(\frac{\sqrt{s_0}}{\sqrt{d}})$.

Proof. Let $W \in \text{Mat}_{d\times d}$ be a random square matrix, with entries drawn independently from $\sigma(0, 1/d^2)$. Define the function $N(\vec{a}) = \sum_{i=1}^{d} \text{ReLU}(Wx)_i$. Then $N(\vec{a})$ is a sum of d i.i.d. random variables of the form $N_i = \text{ReLU}(x) \mid x \sim \sigma(0, |\vec{a}|/d)$. Applying arguments similar to those used in the proof of the previous theorem, we see that N_i has norm $c \cdot |\vec{a}|/d$, for $c > 0$ the absolute constant

$$c = \mathbb{E}_{x\sim\sigma(0,1)}\text{ReLU}(x) = \frac{1}{2\sqrt{\pi}}.$$

The variance of N_i is $O(|\vec{a}|^2)/d$, and N_i is bounded o.n.p. by $\tilde{O}(|\vec{a}|)$. Thus Bernstein's inequality implies that, o.n.p.,

$$N(\vec{a}) = \sum_{i=1}^{d} N_i = c \cdot |\vec{a}| + \tilde{O}(|\vec{a}|/\sqrt{d}).$$

Now $|\vec{a}| < s_0 = \tilde{O}(\sqrt{s})$, so $N(\vec{a}) = |\vec{a}| + \tilde{O}(\varepsilon)$. Let $f(y) = \sqrt{s_0 - y^2}$ (for $|y| \leq \sqrt{s_0}$), a semicircle of radius $\sqrt{s_0}$ viewed as a function of a real variable. Define the piecewise-linear function f_{PL} given by splitting the semicircle into d equal arcs, and connecting the endpoints of the arcs (extending the first and last arc linearly outside the domain of definition). The difference between the values of f on the endpoints of each arc is bounded by its arclength, which is $O(\sqrt{s_0}/d)$. Thus $|f(x) - f_{PL}(x)| = O(\sqrt{s_0}/d)$ (in fact, much better asymptotic bounds are possible.) Now f_{PL} is a sum of d ReLUs, thus it is a scalar-valued function which can be expressed by a width-d neural net. Now choose a random "approximately unit" vector $v \in \mathbb{R}^d$ according to the Gaussian $v \sim \sigma(0, 1/\sqrt{d})$. Now we define the neural net $\text{balance}(\vec{a}) := \vec{a} + f_{PL}(N(\vec{a}))v$. Since both $N(\vec{a})$ and f_{PL} can be expressed as width-d neural nets, balance can be expressed as a width-$O(d)$ neural net. Now since v is a random vector, we have, o.n.p.,

$$v \cdot \vec{a} = \tilde{O}(|\vec{a}|/\sqrt{d})$$

and $v \cdot \vec{\phi}_k = \tilde{O}(\frac{1}{\sqrt{d}})$. Since there are at most polynomially-many (in r) features, the "negligible probablity" exceptions remain negligible when combined over all features. The bound $N(\vec{a}) = \tilde{O}(\sqrt{s})$ thus implies both bounds in the theorem. □

Let $\alpha(x), \beta(x, y)$ be functions. Let W be random and Φ be a matrix of features. Fix $k, \ell \in \{0, \ldots, m-1\}$. Let $\boldsymbol{b} \in \{0,1\}^m$ be a boolean vector. Let $\boldsymbol{b}_{kl} = \boldsymbol{b}_k\vec{\phi}_k + \boldsymbol{b}_\ell\vec{\phi}_\ell$, and $\boldsymbol{b}' = \boldsymbol{b} - \boldsymbol{b}_{k\ell}$. Outside negligible probability, we know that $\Phi(\boldsymbol{b}_{k\ell}) \cdot \Phi(\boldsymbol{b}') = \tilde{O}(\varepsilon)$. This means that if $\varepsilon = \tilde{\Theta}(1/\sqrt{d})$ and we apply a random matrix W then we still have $W\Phi(\boldsymbol{b}') \cdot W\Phi(\boldsymbol{b}_{k\ell}) = \tilde{O}(\varepsilon)$ (outside negligible probability). Define $\vec{x}_{kl} = W\Phi(\boldsymbol{b}_{kl})$ and $\vec{x}' = W\Phi(\boldsymbol{b}')$. Since random matrices are $O(d)$-invariant, we can assume WLOG that these are drawn independently and randomly from appropriate Gaussian distributions *EXPAND*. Specifically, $\vec{x}_{kl}$ is drawn from a distribution with variance 2 and $\vec{x}'$ is drawn from a distribution with variance $O(s)$.

Define

$$\mathcal{M}_w(\vec{x}) = \alpha(W(\vec{x})),$$

and define

$$R^i_{k\ell} := \beta(\vec{\phi}^i_k, \vec{\phi}^i_\ell).$$

Lemma 20. *For suitable choices of a piecewise-linear function α and some function β (both depending on s) we can guarantee that $R_{k,\ell} \cdot \mathcal{M}_w(\Phi(\boldsymbol{b})) = \boldsymbol{b}_k \wedge \boldsymbol{b}_\ell + \tilde{O}(\varepsilon_{\text{out}})$.*

Proof. As explained above, we can assume that $x^i_{k\ell}, (x^i)'$ are drawn from independent boolean distributions with variance respectively $\frac{2}{d}, \frac{s}{d}$. Define $X = \sigma(0, \frac{s}{d}I)$ to be the Gaussian variable with variance $\frac{s}{d}$. Define

$$\Delta_i(x) := \alpha\left(x + \vec{x}^i_{k\ell}) - \alpha(x)\right).$$

Write

$$\mathcal{M}_{w\Delta}(\vec{y})^i := \Delta_i(\vec{y}).$$

Then $\mathcal{M}_w(\vec{x}) = \mathcal{M}_w(\vec{x}') + \Delta_i(\vec{x}')$. It remains to prove the following sublemma:

Lemma 21. *(Outside negligible probability:)*

$$R_{k\ell} \cdot \mathcal{M}_w(\vec{x}') = \tilde{O}(\varepsilon_{\text{out}}) \tag{8}$$

$$R_{k\ell} \cdot \Delta_i(\vec{x}') = \boldsymbol{b}_k \wedge \boldsymbol{b}_\ell + \tilde{O}(\varepsilon_{\text{out}}) \tag{9}$$

We start with the first expression. We have

- $\vec{x}'_i$ random from Gaussian X, variance s/d.
- $\alpha(\vec{x}'_i)$ random, bounded by B (o.n.p. bound for α on X).
- From POV of x' : we know (x, y) random Gaussian, variance $1/d$.
- So $R_{kl} \cdot \mathcal{M}_w(\vec{x}')$ is the sum of d samples of $\beta(x, y)\alpha(z)$ for x, y, z from appropriate Gaussians.
- WTS: ± symmetric in independent way, variance $\tilde{O}(\varepsilon_{\text{out}})/d$, bounded (onp) by $\tilde{O}$ of stdev (check if this bound correct for Azuma inequality). For this (modelling on quadratic case): choose β to be ± symmetric in either coordinate independently, and appropriately bounded.

For the second expression, we treat two cases, namely $(\boldsymbol{b}_k, \boldsymbol{b}_\ell) \in \{(1, 1), (0, 1)\}$. We do not need to treat other cases as $(1, 0)$ follows by symmetry and $(0, 0)$ is trivial. Start with $(1, 1)$ case, so $\boldsymbol{b}_k \wedge \boldsymbol{b}_\ell = 1$. We then have

- Want
$$E\left(\beta(x, y)\Delta_{x,y}(z))\right)$$
to be 1.
- Above bounded to make Azuma ok (prob enough to check $\Delta = O(1)$ and use Azuma bounds from previous).

Final case, $(0, 1)$.

- Want $E((\beta(x, y)\Delta_x(z)))$ to be 1.
- This follows from ± symmetry of β (and bounds as above).

□

E.4. Error correction layers

Theorem 22. *Suppose we are in the context of Appendix A. Then there exists a polylog constant $K = K(d)$ and a single-layer neural net $\mathcal{M}_w(x) = v_1 + W_1\text{ReLU}(v_0 + W_0(x))$ and a feature matrix $\Phi^{(1)} \in \text{Mat}_{d\times m}$ such that if $\varepsilon(= \varepsilon^{(0)}) < K\frac{d^{1/4}}{m^{1/2}s^{1/4}}$, then for each input x, o.n.p., the feature $\vec{\phi}^{(1)}_k$ linearly separates the boolean function f_k on the activation $\vec{a}^{(1)}(x) = \mathcal{M}_w(x)$, with error*

$$\varepsilon^{(1)} = O\big(\log(d) \cdot \frac{\sqrt{s}}{\sqrt{d}.}\big)$$

Moreover, we can choose the new feature vectors $\phi^{(1)}_k$ such that they have feature interference bounded by

$$\mu^{(1)} = \tilde{O}\big(\frac{\sqrt{s}}{\sqrt{d}}\big).$$

Proof. We begin by defining an unnormalized version of the output feature matrix. Define $p = \frac{1}{\sqrt{ds}}$, a probability parameter. Let $\Phi^{(1),u} \in \text{Mat}_{m\times d}$ be a matrix of entries M^i_k drawn uniformly from the ternary random variable $\begin{cases} p(M^i_k = 1) & = p/2 \\ p(M^i_k = -1) & = p/2 \\ p(M^i_k = 0) & = 1 - p \end{cases}$.

Let $\Gamma \subset \{0, \ldots, m\} \times \{0, \ldots, d\}$ be the set of nonzero values of $\Phi^{(1),u}$. Note that (o.n.p.), it has size

$$|\Gamma| = m\sqrt{\frac{d}{s}} + \tilde{O}(1).$$

We think of this as a graph, connecting each feature k to a set of (approximately $\sqrt{\frac{d}{s}}$) neurons it "activates", $\Gamma_k \subset \{1, \ldots, d\}$. We also write $\Gamma_i \subset \{1, \ldots, m\}$ for the set of features connected to the ith neuron.

Let

$$\mathrm{round}_{[0,1]}(x) := 3\left(\mathrm{ReLU}(x - 1/3) - \mathrm{ReLU}(x - 2/3)\right),$$

the piecewise-linear function that maps $\mathbb{R}$ to the interval $[0, 1]$ and is non-constant only on the interval $(1/3, 2/3)$.

Now for any integer, define

$$\mathrm{round}_{[0,a]}(x) := \mathrm{round}_{[0,1]}(x) + \mathrm{round}_{[0,1]}(x - 1) + \cdots + \mathrm{round}_{[0,1]}(x - a + 1),$$

and similarly,

$$\mathrm{round}_{[-a,a]}(x) := \mathrm{round}_{[0,a]}(x) - \mathrm{round}_{[0,1]}(-x).$$

This is a piecewise-linear "staircase" function with the following properties:

- $\mathrm{round}_{[-a,a]}(x) \in [-a, a]$ for all $x \in \mathbb{R}$ and
- $\mathrm{round}_{[-a,a]}(n + \varepsilon) = n$, whenever $n \in [-a, a]$ is an integer and $\varepsilon < 1/3$.

Thus for all sufficiently small values x, the function round will "round" x to the nearest integer, so long as the nearest integer is less than $1/3$ away; hence its name. By construction, the function $\mathrm{round}_{[-a,a]}(x)$ is a sum of a $4a$ ReLUs.

We will use for our nonlinearity the function

$$\mathrm{round}(x) = \mathrm{round}_{[-2,2]}(x):$$

$$x(x)$$

(Using larger intervals $[-a, a]$ in our nonlinearity $\mathrm{round}_{[-a.a]}$ would give slightly stronger results, but won't be needed.)

Now we define the unnormalized neural net model as follows:

$$\mathcal{M}_w{}^u(x) := \mathrm{round}(\Phi^{(1),u}\left(\Phi^{(0)}\right)^T(x)). \tag{10}$$

Finally, we normalize:

$$\mathcal{M}_w(x) := \frac{\mathcal{M}_w{}^u(x)}{\sqrt{d/s}} \tag{11}$$

$$\Phi^{(1)} := \frac{\Phi^{(1),u}}{\sqrt{d/s}}. \tag{12}$$

For each feature $k \in \{1, \ldots, m\}$ in an input x, the unnormalized neural net $\mathcal{M}_w{}^{(1),u}$ roughly does the following.

1. "Reads" the feature ϕ_k
2. "Writes" 1s in all neurons $i \in \Gamma_k$ connected to k assuming ϕ_k is present
3. "Rounds" each neuron which is close to $-2, -1, 0, 1$ or 2 to the closest integer.

At the end, we hope to obtain a vector with exactly the entry $M_k^i \in \pm 1$ for each k with $f_k(x) = 1$ and zero elsewhere. If we're lucky and there are no issues with excess interference and no pairs of active features k, ℓ that share a neuron $i \in \Gamma_k \cap \Gamma_\ell$, the result of this computation will be $\Phi^{(1),u}(\boldsymbol{b}(x))$, and its error can then be controlled by understanding the interference of the new normalized feature matrix $\Phi^{(1)}$.

In order to make this work, we need to control two types of issues:

- *Collision*: it's possible that two simultaneously active features k, ℓ with $f_k(x) = f_\ell(x) = 1$ share some neurons, so some of the entries of $\vec{\phi}_k^{(1),u} + \vec{\phi}_\ell^{(1),u}$ have "colliding" information from the kth and ℓth neurons that gives the wrong answer after getting rounded to one of $\{-2, -1, 0, 1, 2\}$.
- *Interference*: it's possible that, even if Γ^k are disjoint for all features k appearing in $\boldsymbol{b}(x)$, the various interference terms shift the value far enough from the "correct" value in $\{-1, 0, 1\}$ that the "round" function does not successfully return it to its original position.

These are controlled by the two parts of the following lemma.

Lemma 23. *1. For any $x \in X$, we have o.n.p.:*

$$|| \left(\Phi^{(1,u)} \left(\Phi^{(0)} \right)^T (x) - \Phi^{(1),u} \boldsymbol{b}_x \right) ||_\infty = o(1).$$

2. For any boolean $\boldsymbol{b}$ with sparsity $||\boldsymbol{b}||_1 < s$, we have (o.n.p.) the difference

$$\overrightarrow{\text{err}}_{\textit{collision}} := \text{round}(\Phi^{(1,u)}(\boldsymbol{b})) - \Phi^{(1,u)}(\boldsymbol{b}) \in \mathbb{R}^d$$

has all unnormalized feature readoffs

$$\vec{\phi}_k^u \cdot \overrightarrow{\text{err}}_{\textit{collision}} = \tilde{O} \max(1, \sqrt{s^3/d}).$$

Proof. Note that the two results are both about ℓ_∞ errors, but in two different spaces, namely in the space $\mathbb{R}^d$ with the neuron basis for part (1) and in the space $\mathbb{R}^m$ with the feature basis for part (2). We start with part (1). Since there is a polynomial number of neurons, bounding the ℓ_∞ error o.n.p. is equivalent to bounding the difference for each coordinate:

$$E_i(x) := \left(\Phi^{(1,u)} \left(\Phi^{(0)} \right)^T (x) - \Phi^{(1),u} \boldsymbol{b}(x) \right) \cdot \vec{\boldsymbol{e}}_i.$$

This difference is a linear combination of the errors $\vec{\phi}_k^{(0)} \cdot x$, with coefficients given by the matrix coefficients $\left(\Phi^{(1,u)}\right)_i^k$, with i fixed and k varying. For a pair $(i, k) \in \Gamma$, let $\sigma(i, k) \in \pm 1$ be the sign of the corresponding matrix coefficient (which is chosen independently at random in the random variable-valued definition of our neural net). We then have

$$E_i(x) = \sum_{k \in \Gamma^i} \sigma(i, k) x \cdot \vec{\phi}_k^{(0)}.$$

By assumption, $x \cdot \phi_k \le \varepsilon^{(0)}$. Since the signs are chosen independently at random, we can bound this value o.n.p. by the Bernstein inequality, Theorem 32, with discrete variables $X_k = \sigma_{i,k} x \cdot \vec{\phi}_k^{(0)}$. Here k is indexed by a $|\Gamma^i|$-element set. By definition of $\Phi^{(1)}$, each element $\{1, \ldots, m\}$ has probability $p = \frac{1}{\sqrt{sd}}$ of being in Γ^i, so

$$|\Gamma^i| = \frac{m}{\sqrt{sd}} + \tilde{O}\left(\frac{\sqrt{m}}{\sqrt{sd}} \right) = \tilde{O}\left(\frac{m}{\sqrt{sd}} \right).$$

Since all these random variables are bounded by $\varepsilon^{(0)}$ in absolute value, Bernstein's inequality implies that o.n.p.,

$$E_i = O\left(\varepsilon^{(0)} \cdot \left(\frac{m}{\sqrt{sd}} \right)^{1/2} \right),$$

giving part (1) of the lemma.

To prove the second part, note that the "ground truth" activation $\vec{a}_{\text{ground}} := \Phi^{(1,u)}\boldsymbol{b}$ is an integer-valued vector with coefficients $(\vec{a}_{\text{ground}})_i = \sum_{k\in\Gamma^i\cap\boldsymbol{b}}\sigma_k$. It is changed by applying the round function if and only if this sum is > 2 in absolute value, i.e., if it is a "collision" (i.e., contained in the intersection) of at least 3 subset of the form Σ_k. The expectation of the number of such overlaps a given neuron $i \in \{1, \dots, d\}$ can be can be bounded by

$$O(\frac{s^3}{(\sqrt{sd})^3}) = O\left(\frac{s^{3/2}}{d^{3/2}}\right).$$

Thus the coefficients of the error vector

$$(\overrightarrow{\text{err}}_{\text{collision}})_i = (\vec{a}_{\text{ground}})_i - \mathrm{round}(\vec{a}_{\text{ground}})_i$$

are drawn i.i.d. from a distribution with mean 0 (as it is symmetric) and variance bounded by $\tilde{O}(\frac{s^{3/2}}{d^{3/2}})$, which is absolutely bounded by $\tilde{O}(1)$. In other words, we have o.n.p. that this vector has at most

$$\tilde{O}\max\left(1, \left(s^{3/2}\sqrt{d}\right)\right)$$

entries all bounded by $\tilde{O}(1)$, and with independently random signs. When we take the dot product with another unnormalized feature vector we are left with an error bounded by

$$\varepsilon_{\text{collision}} \cdot \vec{\phi}_k^{(1),u} = \tilde{O}\max\left(1, \left(\frac{s^{3/2}}{d^{1/2}}\right)\right),$$

completing the proof. □

Now we can finish the proof. The interference bound in the lemma implies that o.n.p., the d-dimensional vector

$$\Phi^{(1),u}(\boldsymbol{b}) - \Phi^{(1),u}(\Phi^{(0,T)}(x))$$

has all coefficients bounded by $o(1)$, an in particular, bounded by $1/3$. Since the LHS has all integer entries, this means that

$$\mathrm{round}(\Phi^{(1),u}(\boldsymbol{b})) = \mathrm{round}(\Phi^{(1),u}(\Phi^{(0,T)}(x)))$$

(As the "round" function is constant on $[n - 1/3, n + 1/3]$ for any integer n).

Since we have assumed that $s < d^{1/3}$ (in A), the asymptotic term $s^{3/2}/d^{1/2}$ in the collision error bound is bounded by 1, so o.n.p., $\varepsilon_{\text{collision}} \cdot \vec{\phi}_k^{(1),u} = \tilde{O}(1)$. Finally, when we normalize, both sides of the dot product get multiplied by $A = s^{1/4}/d^{1/4}$, and so after normalizing the coresponding bound gets multiplied by $\sqrt{s}/\sqrt{d}$, and we get the expression (o.n.p.):

$$\vec{\phi}_k^{(1)} \cdot \left(\mathcal{M}_w(x) - \Phi^{(1)}(\boldsymbol{b}(x))\right) = \tilde{O}(\sqrt{s}/\sqrt{d}).$$

Finally, by a similar argument to the collision proof, we see that the unnormalized dot product $\Phi^{(1),u}(\boldsymbol{b}(x)) \cdot \vec{\phi}_k$ is $\sqrt{d}/\sqrt{s}$ up to an error of $\tilde{O}(1)$, so the error m

We claim that the pair $(\mathcal{M}_w, \Phi^{(1)})$ satisfies (o.n.p.) the conditions for the error-correction circuit above, for some appropriate relationships between the values $d, \varepsilon^{(0)}, \varepsilon^{(1)}$ depending on m, satisfying asymptotic inequalities of the form

$$\varepsilon^{(0)} = \tilde{O}(\frac{d^{1/4}}{m^{1/2}s^{1/4}}),$$

$$\varepsilon^{(1)} = \tilde{O}\left(\frac{\sqrt{s}}{\sqrt{d}}\right),$$

$$\mu^{(1)} = \tilde{O}\left(\frac{\sqrt{s}}{\sqrt{d}}\right).$$

Lemma 24. *For a suitable choice of ε_{in} as above we can guarantee that:*

1. *If $\overrightarrow{\text{err}} \in \mathbb{R}^m$ has $||\overrightarrow{\text{err}}||_\infty < \varepsilon_{\text{in}}$, then $||\Phi(\overrightarrow{\text{err}})||_\infty = o(1)$, o. n. p. (Note that the latter value is an ℓ^∞ norm in the neuron basis.)*

2. *If $\boldsymbol{b}$ is boolean and s-sparse, then $\frac{\Phi}{A}(\text{round}(\Phi(\boldsymbol{b})) \approx_{\varepsilon_{\text{out}}} \boldsymbol{b}$, o. n. p.*

To get part (1) above, observe that for any neuron index i, we have

$$\Phi(\overrightarrow{\text{err}})_i = \sum_{k | k \in \Gamma^i} \sigma_{k,i} \overrightarrow{\text{err}}_k,$$

where we define $\Gamma^i := \{k \mid (k, i) \in \Gamma\}$. Since the signs $\sigma_{k,i}$ are random and independent, this is a sum with random signs of numbers of absolute value $< \varepsilon_{\text{in}}$. From the Azuma inequality, we see that (o. n. p.) $\Phi(\overrightarrow{\text{err}})_i = \tilde{O}(\overrightarrow{\text{err}} \cdot \sqrt{|\Gamma_i|})$. Since the Γ was chosen randomly, o. n. p.

$$|\Gamma_i| = \tilde{\Theta}(|\Gamma|/d) = \tilde{\Theta}(d^{\frac{1-\gamma}{2}}) = o(\varepsilon_{\text{in}}^{-2}).$$

The last statement follows from comparing exponents in the two sides, and the freedom of choice of polylog term in ε_{in}.

For part (2) above, observe that $(\mathbf{R}_{\text{out}}(\text{round}(\Phi(\boldsymbol{b}))))_k$ is the average over the set $\Gamma_k = \{i \mid (k, i) \in \Gamma\}$ of

$$a_i := \text{round}\left(\sum_{\ell \in S} \Phi_{\ell,i}\right)$$

where S is the set of features that are on in $\boldsymbol{b}$, of size $|S| \leq s$. We want to compare this to $\boldsymbol{b}_k$, which is 1 if $k \in S$ and 0 otherwise. We expect (for $i \in \Gamma_k$) that $a_i = 0$ if $\boldsymbol{b}_k = 0$ and $a_i = 1$ if $\boldsymbol{b}_k = 1$. Since $\text{round}()$ always returns a value of absolute value ≤ 1, we can bound the error by twice the number of incorrect values. We get errors of two types.

1. *Interference error*, from neurons that are on when they should be off. I.e., when $a_i \neq 0$ despite $\boldsymbol{b}_k = 0$.

2. *Collision error*, from neurons which should be on but are 0 (or have wrong sign) due to contributions from both S_k and another feature.

Either of these errors happens when Γ_k and $\bigcup_{\ell \in S'} \Gamma_\ell$ intersect for $S' = S \setminus \{k\}$, the set of nonzero values of $\boldsymbol{b}$ not equal to k. Now Γ_k has $\tilde{O}(d^{\frac{1-\gamma}{2}})$ nonzero entries and $\bigcup_{\ell \in S'} \Gamma_\ell$ has at most $\tilde{O}(d^{\gamma + \frac{1-\gamma}{2}})$ entries; since each subset Γ_k is independently random, we see (o. n. p.) that the intersection has at most $\tilde{O}(\frac{d^{\frac{1+\gamma}{2}} d^{\gamma + \frac{1+\gamma}{2}}}{d}) = \tilde{O}(1)$ entries, and the average is indeed $\tilde{O}(\varepsilon_{\text{out}})$.

This completes the proof of the lemma. The theorem follows. Indeed, suppose that $x \in \mathbb{R}^{d\text{in}}$ is a vector with $\mathbf{R}_{\text{in}}(x) \approx_{\varepsilon_{\text{in}}} \boldsymbol{b}$ for $\boldsymbol{b} \in \{0, 1\}^m$ an s-sparse boolean vector. Setting $\overrightarrow{\text{err}} = \mathbf{R}_{\text{in}}(x) - \boldsymbol{b}$, part (1) implies that

$$\Phi \circ \mathbf{R}_{\text{in}}(x) - \Phi(\boldsymbol{b})$$

has coefficients at most $o(1)$; since $\Phi(\boldsymbol{b})$ has integer entries, this means that applying round to both sides produces the same results. Part (1) then implies that the RHS $\Phi(\boldsymbol{b})$ has sufficiently small interference. □

Corollary 25 (Lemma 7)**.** *For sufficiently small input interfefrence there exists a 1-layer MLP that returns (outside negligible probability) an encoding of the same boolean vector with low interference ($1/\sqrt{d}$ assuming low sparsity parameter).*

Proof. This follows from the theorem in the case $\gamma = 0$, i.e., when the sparsity parameter s is polylog in m. □

E.5. Targeted superpositional AND *likely remove*

In this section we prove a generalization of Theorem 3.2.

Let $\Gamma \subset \{1, \ldots, m\}^{[2]}$ be the edges of a graph (here the superscript [2] denotes the "exterior power" of a set, i.e., the set of $\binom{m}{2}$ unordered pairs). Assume that the number of edges $|E_\Gamma| = \tilde{O}(m)$. Let $\mathcal{C}_\Gamma : \{0,1\}^m \to \{0,1\}^{E_\Gamma}$ be the circuit with value

$$\mathcal{C}_\Gamma(\boldsymbol{b})_{(k,\ell)} = \boldsymbol{b}_k \wedge \boldsymbol{b}$$

at the unordered pair $(k, \ell) \in E_\Gamma$ corresponding to an edge of Γ. We think of $\mathcal{C}_\Gamma$ as the (not quite universal) circuit that takes AND's of pairs of features in Γ and returns a boolean vector of roughly the same size.

We will show that this circuit can be emulated with suitably small interference on the output.

The proof is very similar to the proof of the error correction theorem above (Theorem **??**), in particular with the main argument controlled by a subset $\Gamma \subset \{1, \ldots, m\} \times \{1, \ldots, d\}$, with m the number of *edges* of Γ (i.e., outputs of the circuit).

There are however two main differences.

1. What we read from each subset $\Gamma_{k,\ell}$ associated to an edge $(k, \ell) \in \Gamma$ is a the result of a nonlinearity applied to a *sum* of two random ± 1 vectors $\vec{\phi}_k, \vec{\phi}_\ell$ (associated to the two inputs k, ℓ), that returns (up to small error) the sum of neurons in of Γ_{ij} where the signs of $\vec{\phi}_k$ and $\vec{\phi}_\ell$ are both 1.

2. To control interference issues, we need to carefully partition the graph Γ into pieces with a certain asymptotic "balanced" property (see Lemma **??**).

3. The output interference is $\tilde{O}(\sqrt{\frac{s^2}{d}}$ instead of $\tilde{O}(\sqrt{\frac{s}{d}}$ since there are $O(s^2)$ active output features (corresponding to pairs of features that are on).

Theorem 26 (Targeted superpositional AND). *Let m be an integer and $\Gamma \subset \{1, \ldots, m\} \times \{1, \ldots, m\}$ a graph. Assume we have a readoff matrix $\mathbf{R}_{\text{in}} \in \text{Mat}_{m \times d}$ that maps a d-dimensional space to an m-dimensional space, and let $s = o(\sqrt{m})$ be a sparsity parameter (either polynomial or polylogarithmic in m). Let ε_{in} be an interference parameter.*

Assume that we have $\varepsilon_{\text{in}}^2 md\sqrt{d/s} = \tilde{O}(1)$ is bounded by some sufficiently small inverse polylogarithmic expression in m. Then there exists a single-layer mixed emulation $\mathcal{M}_w(x) = \text{ReLU}(W_{in}x + w_{bias})$ of the universal AND circuit $\mathcal{C}_{\text{uand}}$ (together with an "output readoff" matrix $\mathbf{R}_{\text{out}}$) such that $\mathcal{M}_w$ is an emulation of $\mathcal{C}_\Gamma$ on the input class $\mathcal{B} = \mathcal{B}_s$ of boolean vectors of sparsity $\leq s$, with precision $\varepsilon_{\text{in}} \to \varepsilon_{\text{out}}$, for $\varepsilon_{\text{out}} = \tilde{O}\left(\sqrt{\frac{s^2}{d}}\right)$.

Before proving the theorem, we note that our UAND statements are corollaries:

Corollary 27 (U-AND with basis-aligned inputs). *Fix a sparsity parameter $s \in \mathbb{N}$. Then for large input length m, there exists a single-layer neural network $\mathcal{M}_w(x) = \text{MLP}(x) = \text{ReLU}(W_{in}x + w_{bias})$ that ε-linearly represents the universal AND circuit $\mathcal{C}_{\text{UAND}}$ on s-sparse inputs, with width $d = \tilde{O}_m(1/\varepsilon^2)$ (i.e. polylogarithmic in m).*

This follows from the fact that the incoming interference $\varepsilon_{\text{in}} = 0$ since the incoming feature basis is basis-aligned.

Corollary 28 (U-AND with inputs in superposition). *Let $s \in \mathbb{N}$ be a fixed sparsity limit and $\varepsilon < 1$ a fixed interference parameter. There exists a feature encoding Φ and single-layer neural net $\mathcal{M}_w(x) = \text{MLP}(x) = \text{ReLU}(W_{in}x + w_{bias})$ with input size m_{in} and width $d = \tilde{O}(\sqrt{m_{\text{in}}}/\varepsilon^2)$, such that $\mathcal{M}_w \circ \Phi$ ε-linearly represents $\mathcal{C}_{\text{UAND}}$ on all s-sparse inputs $\boldsymbol{b}$.*

This follows by restricting all but $m_{\text{in}} = \sqrt{m}$ input features to 0 and taking Γ to be the complete graph on vertices $\{0, \ldots, m_{\text{in}}\}$.

Now we prove the theorem.

Proof. We begin by considering a simpler case. We say that a graph Γ with m edges is *self-balanced* if each vertex has degree at most $\tilde{O}(1)$ (some fixed polylogarithmic-in-m bound).

Suppose Γ is self-balanced. Define $A := \sqrt{d/s}$. For each edge $(k, \ell) \in \Gamma$, choose at random a subset $\Gamma_{k\ell} \subset \{1, \ldots, d\}$ of size within a polylog error of A. Write also

$$\Gamma_k = \bigcup_{\ell | (k,\ell) \in \Gamma} \Gamma_{k,\ell}.$$

Write down feature vectors $\vec{\phi}_{k\ell} = \sum_{i \in \Gamma_k} \pm \vec{e}_i$, with signs $\sigma_{k,i}$ chosen independently and randomly for each k, i. For a pair $k, \ell \in \Gamma$, define the vector $\vec{r}_{k,\ell}$ to be the indicator of the set of neurons

$$\Gamma^{\text{out}}_{k,\ell} := \{i \in \{1, \ldots, d\} \mid \sigma_{k,i} = \sigma_{\ell,i} = 1\},$$

Note that $|\Gamma^{\text{out}}_{k,\ell}|$ has, o. n. p., within a polylog difference from $\frac{1}{4}|\Gamma_{k,\ell}| = \frac{A}{4}$ elements.

Write

$$\vec{\phi}^{\text{in}}_k := \sum_{\ell | (k,\ell) \in \Gamma} \vec{\phi}_{k,\ell},$$

Note that this is a indicator function of a union polylog-many independently chosen sets of size A. Write Φ^{in} for the $m \times d$ matrix with columns $\vec{\phi}^{\text{in}}_k$.

Now we define the emulation net to be

$$\mathcal{M}_{w\Gamma}(x) = \frac{4}{A} \text{ReLU}(\Phi^{\text{in}}(x) - 1).$$

We note that (outside interference and collision errors of frequency bounded o. n. p. by $\tilde{O}(\varepsilon_{\text{out}})$,) we have

$$\text{ReLU}(\Phi^T(\boldsymbol{b})) - 1)_i = \begin{cases} 1, & \exists k, \ell \in S \text{ with } i \in \Gamma_{k,\ell} \text{ and } \sigma_{k,i} = \sigma_{\ell,i} = 1 \\ 0, & \text{otherwise,} \end{cases}.$$

Here as before we take $S \subset \{1, \ldots, m\}$ for the set of features that are on.

Analogously to our proof of Lemma 24's part **??** we see that the difference $\Phi^T(\boldsymbol{b}) - \Phi^T(\mathbf{R}_{\text{in}}(x))$ is (o. n. p.) bounded by $o(1)$, and thus we are done just as in the previous lemma.

For general graphs Γ, we might have an issue if some vertices have very high degree; if one were to try to run the same proof, their corresponding features would then admit unmanageably high interference.

To fix this, we note that in order to emulate $\mathcal{C}_\Gamma$ it is sufficient (up to polylogarithmically increasing the number of neurons) to emulate $\mathcal{C}_{\Gamma_1}, \ldots, \mathcal{C}_{\Gamma_T}$ for some polylogarithmic collection of graphs Γ_t with $\cup_t \Gamma_t = \Gamma$. We now split an arbitrary graph Γ into subgraphs with a nice "balanced" property.

Let $a, b \in \mathbb{R}$ be parameters. We say that a graph is a, b-balanced if it is bipartite on a pair of disjoint subsets of vertices $V_0, V_1 \subset \{0, \ldots, m\}$, such that $|V_0| = a$, $|V_1| = b$ and each vertex in V_0 has degree at most m/a and each vertex in V_1 has degree at most m/b. We say a graph $\Gamma \subset \{0, \ldots, m\}^{[2]}$ is balanced if it is a, b-balanced for some a, b.

It can be shown using an inductive argument that any graph Γ with m edges can be written as a union of $\text{polylog}(m.)$

Now it remains to show that the theorem holds for a balanced graph. Indeed, suppose that Γ has vertices supported on $V_0 \sqcup V_1 \subset \{1, \ldots, m\}$ and is a, b-balanced. Suppose (WLOG) that $a \leq b$. Then we randomly partition the neurons $\{1, \ldots, d\}$ into a roughly equal sets Γ_k for $k \in V_0$ (equivalently, we choose a random map $\{1, \ldots, d\} \to V_0$ and define Γ_k to be the preimage of k). We then choose for $\ell \in V_1$ the set $\Gamma_{k,\ell}$ to be a random subset of size about $\sqrt{d/s^2}$ inside Γ_k, and define $\Gamma_\ell = \cup_{k|(k,\ell) \in \Gamma}$. We finish the argument by bounding the errors in the same way as in the self-balanced case, concluding the proof. □

F. Theoretical Framework and Statistical Tools

Here we provide statistical definitions and lemmas required for our proofs in Appendix E.

F.1. Negligible probabilities

Most results in this paper are proven *outside negligible probability*. This is a standard notion in complexity theory and cryptography (Bellare, 2002), with the following formal definition:

Definition 8. *Let $\{E_n\}_{n=1}^{\infty}$ be a sequence of events parameterized by n. We say that E_n is true* ***with negligible probability*** *(w. n. p.) if for any polynomial exponent $c \in \mathbb{N}$, there exists some constant $N_c \in \mathbb{N}$ such that $P(E_n) < O(n^{-c})$ for all $n > N_c$. Similarly, we say that E_n is true* ***outside negligible probability*** *(o. n. p.) if its complement $\overline{E_n}$ is true with negligible probability.*

If $E_n = E_n(x)$ depends on an input in some set X, when we say $E_n(\boldsymbol{b})$ is true with negligible probability for all fixed inputs $x \in X$ we implicitly assume that there is an explicit constant $C_n < O(n^{-c})$ as above that bounds the probability of $E_n(x)$ for each valid input $x \in X$.

Intuitively, the reason why this probability is "negligible" is that the union of polynomially many events of negligible probability also has negligible probability. As we never consider more networks requiring more than polynomially many operations, we can ignore events of negligible probability at each step when performing asymptotic analysis, which greatly simplifies our proofs.

Example 2. Let $\boldsymbol{b}$ be a random boolean vectors of length n. Then outside negligible probability, $\boldsymbol{b}$ has between $n/2 + \log(n)\sqrt{n}$ and $n/2 - \log(n)\sqrt{n}$ zeroes.

This follows from the central limit theorem. (Note that if we used $\sqrt{\log(n)}\sqrt{n}$, the result would be false!)

For cases where the event is a bound on a random function (as above), we can combine "negligible probability" notation and big-O, as well as big-$\tilde{O}$ notation, as follows.

Definition 9. *Suppose a function $f(x) = f_n(x)$ depends on the complexity parameter n and a fixed input $x \in X$ and is valued in random variables*[3]*. Let $g(x) \geq 0$ be a deterministic function*[4]*. Then we say that*

$$f(x) = \tilde{O}(g(x))$$

if there exists a polylog constant $K_n = O(polylog(n))$ such that, for any input x, the event $|f(x)| < g(x)K(x)$ is true outside negligible probability.

This lets us rephrase the previous example as "for $\boldsymbol{b}$ a random boolean vector of length m, we have $\sum \boldsymbol{b}_k(x) = m/2 + \tilde{O}(\sqrt{m})$." We also list the following result, which will be important for us.

Lemma 28. *Let $d \in \mathbb{N}$ be a complexity parameter. Let $v \in (0, \Gamma/d)$ be a Gaussian-distributed random vector in $\mathbb{R}^d$, and let $x \in \mathbb{R}^d$ be a fixed input vector. Then, outside negligible probability, we have*

1. $|v| = 1 + \tilde{O}\left(\frac{1}{\sqrt{d}}\right)$
2. $v \cdot x = \tilde{O}\left(\frac{|x|}{\sqrt{d}}\right)$.

Proof. The first statement is standard (and follows from the central limit theorem applied to the real variable $(0,1)^2$). The second statement follows from the fact that sums of Gaussian random variables are Gaussian (and variance adds). □

Note that this in particular implies a version of the Johnson-Lindenstrauss lemma:

Corollary 29. *Suppose m is a polynomial function of d (which we take to be the complexity parameter), and suppose $\vec{\phi}_1, \ldots, \vec{\phi}_m \in \mathbb{R}^m$ are random vectors drawn from $(0, \Gamma/d)$. Then outside negligible probability,*

$$\vec{\phi}_k \cdot \vec{\phi}_\ell = \begin{cases} 1 + \tilde{O}(1/\sqrt{d}), & k = \ell \\ \tilde{O}(1/\sqrt{d}), & k \neq \ell. \end{cases}$$

[3] The input can be an "empty input", i.e., f is itself a random variable depending only on n

[4] or a constant depending on n if x is an empty input

Proof. We are checking polynomially many (namely, $O(m^2)$ with m polynomial in d) statements, thus by the union bound, it suffices to show that each is true outside negligible probability. The corollary now follows by inductively on k applying 28 to $\vec{\phi}_k \cdot \vec{\phi}_k = |\vec{\phi}_k|^2$ and $\vec{\phi}_k \cdot \vec{\phi}_\ell \mid \ell < k$, in the latter case taking the vector $\vec{\phi}_\ell$ as fixed. □

Before continuing, we record the following simple result, which will allow us to convert "negligible probability" results to our existence results in the body.

Theorem 30. *Suppose that $s = O(1)$ is a constant sparsity parameter, $\mathcal{M}_w$ is a model in a fixed class that depends on some random parameters, and a property $P(\mathcal{M}_w, x)$ holds outside negligible probability for all inputs $x = \Phi(\boldsymbol{b})$ corresponding to boolean inputs $\boldsymbol{b} \in \{0,1\}^m$ of sparsity s. Then there exists a model $\mathcal{M}_w$ such that the property $P(\mathcal{M}_w, \boldsymbol{b})$ holds for all boolean inputs $\boldsymbol{b}$.*

Proof. This follows from the union bound, since the number of possibly inputs $\boldsymbol{b}$ with sparsity s is $\binom{m}{s} < m^s$ (and negligible probability goes to zero faster than any inverse polynomial). □

Remark 31. For every "negligible probability" statement we encounter, it is straightforward to check that, up to decreasing the asymptotic parameters in appropriate $\tilde{O}$-asymptotic assumptions in the variables involved, we can guarantee for a stronger statement to hold: namely, for any fixed c, we can guarantee that the negligible probability p asymptotically satisfies $p = O(\exp(-\log(m)^c))$. Thus (by another union bound), statements that are true with negligible probability for any boolean input $\boldsymbol{b}$ of size $||\boldsymbol{b}||_1 = \tilde{O}(1)$ (at most polylogarithmic in m) can be made to hold for all such parameters $\boldsymbol{b}$, for an appropriate choice of parameters.

F.2. Concentration inequalities

Concentration inequalities (in the sense we use here) bound tail probabilities of sums of random variables which are either i.i.d. or "close to" i.i.d. in some sense. As we only care about $\tilde{O}$-type precision in our error bounds (i.e., up to polylog factors) and we need statements to be true only outside negligible probability, we are able to get away with very weak versions of bounds which exist in general with much more precision; both of the results we need follow from the Bernstein inequality for martingales (which subsumes the Azuma inequality).

Theorem 32 (Coarse Bernstein bound)**.** *Suppose that $X_1, \ldots, X_n$ are a real random variable bounded by a constant M, which are either i.i.d. or form the difference sequence of a Martingale, i.e., $\mathbb{E}(X_i \mid X_1, \ldots, X_{i-1}) = 0$. Then*

$$\sum x_i = n\mu + \tilde{O}(M\sqrt{n})$$

outside negligible probability, uniformly in the X_i. In other words, there exists a polylogarithmic sequence of constants $K_n = O(polylog(n))$ such that the probability

$$P\big(|\sum_{i=1}^{n}(x_i - [X_i])| < K_n \cdot M\sqrt{n}\big) \leq P_n$$

for some sequence P_n that goes to zero faster than any polynomial function in n.

Proof. This follows from Bernstein's theorem, (**?**). In fact, both statements also follow from the simpler Azuma-Hoeffding inequality. □

Corollary 33. *Let $V = \mathbb{R}^a$ be a vector space, with $a = O(1)$ a constant (we will use $a = 1$ and $a = 3$). Let $\Sigma \in \mathrm{Mat}_{a\times a}$ be a fixed symmetric positive-definite matrix, with $X = (0, \Sigma)$ the corresponding distribution. Let $f : V \to \mathbb{R}$ be a fixed function with subpolynomial growth in x, and let $\mu = [f(x), x \sim X]$ be the mean of f on x drawn from this distribution. Let $x_1, \ldots, x_m$ be a collection of variables drawn from i.i.d. copies of $(0, \Sigma)$. Then o.n.p., $\sum f(x_i) = m\mu + \tilde{O}(\sqrt{m})$, where the polylogarithmic constant in $\tilde{O}$ depends on f.*

Proof. Since f has polynomial growth, $f(x) < K(1+|x|^c)$ for some constants C, d. Thus o.n.p. in m, $f(x) \leq (c\log(Km))$ (note that $f(x)$ doesn't depend on m; we're just saying that $P\big(f(x) \leq d\log(m)\big)$ goes to 0 faster than any polynomial function in m; in fact this probability is $O(m^{-\log(m)})$). Let $M = c\log(Km)$. Then the concentration theorem above implies that

$$\sum_{i=1}^{m}(f(x_i) - [f(x_i)]) = \tilde{O}(M) = \tilde{O}(1),$$

since $M = \tilde{O}(1)$. □

F.3. Precise and mixed emulations

The parameters in the models $\mathcal{M}_w$ in the proofs of our emulation results depend on random matrices of ±1's and 0's, hence can be understood as suitable random variables. In terms of this point of view, we make the following definition.

Suppose that $\mathcal{C} : \{0,1\}^m \to \{0,1\}^{m'}$ is a boolean circuit with input size m. We always assume that the output size m' and the depth are at most polynomial in m. Let $\mathcal{B} \subset \{0,1\}^m$ be a class of inputs (usually characterized by a suitable sparsity property). Let $\varepsilon < 1$ be an interference parameter.

Definition 10. *An* ε-precise emulation of $\mathcal{C}$ *(on input class* $\mathcal{B}$*) is a triple of data* $(\Phi, \mathcal{M}_w, \mathbf{R})$ *all possibly depending on random parameters where* $\Phi \in \mathrm{Mat}_{d_{\text{in}} \times m}$ *is a feature matrix,* $\mathbf{R} \in \mathrm{Mat}_{m' \times d_{\text{out}}}$ *is a readoff matrix and*

$$\mathcal{M}_w : \mathbb{R}^{d\text{in}} \to \mathbb{R}^{d\text{out}}$$

is a (not necessarily linear) function given by a neural net, with the following property:

For any $\boldsymbol{b} \in \mathcal{B}$, *we have, outside negligible probability,*

$$||\mathbf{R} \circ \mathcal{M}_w \circ \Phi(\boldsymbol{b}) - \mathcal{C}(\boldsymbol{b})||_\infty < \varepsilon.$$

Importantly, we do not consider the boolean circuit $\mathcal{C}$ or the input $\boldsymbol{b} \in \mathcal{B}$ to be random variables, and the randomness involved in the negligible probability statement is purely in terms of the parameters that go into the emulation scheme $(\Phi, \mathcal{M}_w, \mathbf{R})$. In particular, this guarantees that if the boolean input $\boldsymbol{b}$ is generated in a non-random way (e.g., adversarially), an emulation nevertheless guarantees (in the "negligible probability sense") safe performance on b so long as the parameters of the emulation were chosen randomly.

It will be useful to extend the notion of emulation to one which correctly approximates $\mathcal{C}$ on inputs $x \in \mathbb{R}^{d\text{in}}$ which represent a boolean input $\boldsymbol{b} \in \{0,1\}^m$ not in the sense of "pure superposition" $x = \Phi(\boldsymbol{b})$ but in the sense of "read-off",

$$||\mathbf{R}_{\text{in}}(x) - \boldsymbol{b}||_\infty < \varepsilon_{\text{in}}.$$

Here $\mathbf{R}_{\text{in}} \in \mathrm{Mat}_{\times m} d_{\text{in}}$ is a readoff matrix that should be thought of as a noisy inverse to the feature matrix on sparse inputs. Formally, we make the following definition. Here we will assume that the matrix $\mathbf{R}_{\text{in}}$ was generated at an earlier stage of the computation, and does not depend on random variables.

Fix a circuit $\mathcal{C} : \{0,1\}^m \to \{0,1\}^{m'}$, a class of inputs $\mathcal{B} \subset \{0,1\}^m$, and an "input readoff" matrix $\mathbf{R}_{\text{in}} \in \mathrm{Mat}_{m \times d_{\text{in}}}$. Let $\varepsilon_{\text{in}}, \varepsilon_{\text{out}}$ be two interference parameters.

Definition 11. *A* mixed emulation of $\mathcal{C}$ with precision $\varepsilon_{\text{in}} \to \varepsilon_{\text{out}}$ *(on input class* $\mathcal{B}$ *and relative to a fixed input readoff matrix* $\mathbf{R}_{\text{in}}$*) is a pair of data* $(\mathcal{M}_w, \mathbf{R}_{\text{out}})$ *both possibly depending on random parameters where* $\mathbf{R} \in \mathrm{Mat}_{m' \times d_{\text{out}}}$ *is a readoff matrix and*

$$\mathcal{M}_w : \mathbb{R}^{d\text{in}} \to \mathbb{R}^{d\text{out}}$$

is a (not necessarily linear) function given by a neural net, with the following property:

For any boolean input $\boldsymbol{b} \in \mathcal{B}$ *and* $x \in \mathbb{R}^{d\text{in}}$ *satisfying*

$$||\mathbf{R}_{\text{in}}(x) - \boldsymbol{b}||_\infty < \varepsilon_{\text{in}},$$

we have, outside negligible probability,

$$||\mathbf{R} \circ \mathcal{M}_w(x) - \mathcal{C}(\boldsymbol{b})||_\infty < \varepsilon_{\text{out}}.$$

Remark 34. Note that if it is impossible to accurately represent $\boldsymbol{b}$ via the matrix $\mathbf{R}$, i.e., to satisfy $||\mathbf{R}(x) - \boldsymbol{b}||_\infty < \varepsilon_{\text{in}}$, then the notion of mixed emulation is vacuous (any neural net would satisfy it for tautological reasons). We will generally apply this notion in contexts where such representations are possible (for example, with via a suitable feature matrix $x = \Phi(\boldsymbol{b})$).

Here as before we do not consider the boolean circuit $\mathcal{C}$ or the input $\boldsymbol{b} \in \mathcal{B}$ to be random variables, and in addition the representation x and the input readoff matrix $\mathbf{R}_{\text{in}}$ are assumed fixed. So the randomness involved in the negligible probability statement is purely in terms of the parameters that go into the pair $(\mathcal{M}_w, \mathbf{R})$.