Mathematical Models of Superposition in Neural Network Architectures

Alexander Julius Busch

December 2025

Abstract

There exists significant empirical evidence that neural networks use *superposition* for representation and computation. Superposition is a representation strategy where networks represent more features than available dimensions in the activation space by leveraging almost-orthogonal directions (directions with small but non-zero dot products). This has been hypothesized to be a key mechanism for the efficiency of large language models and a reason for polysemanticity, where individual neurons represent multiple unrelated features simultaneously, leading to challenges in interpreting neural networks.

In this thesis, we investigate representational and computational superposition in neural networks. For this we mainly follow the influential blog post "Toy Models of Superposition" by Elhage et al. [1], where the authors first illustrated the phenomenon of representational superposition in a simple sparse autoencoder model, and the article "Mathematical Models of Computation in Superposition" by Hanni et al. [2], where the authors investigate random constructions for Boolean computation in superposition.

In this thesis, we aim to

(i) provide a consistent synthesis of foundational definitions and statements regarding features, representations, and superposition in neural networks,
(ii) give precise statements and proofs for the investigation of representational superposition in [1],
(iii) investigate the robustness of emergence of superposition in different settings,
(iv) provide a theoretical and empirical investigation of the polygon solutions found in [1],
(v) give complete derivations and statements of error bounds for the Bernoulli and Gaussian universal And (UAnd) constructions, related to the main text of [2], and

(vi) provide and validate error estimates for neural network constructions modeling the UAnd function.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern neural network architectures have achieved remarkable performance in a broad variety of tasks, including image recognition and language processing. Yet, understanding how these models represent and compute information remains a significant challenge, and neural networks are often regarded as black boxes. Because their parameters are usually optimized starting from random initializations to minimize a complex loss function depending on large datasets, it is not a priori clear whether there is meaningful structure in the parameters and, if such structure exists, whether one can translate it into a useful, human-interpretable form.

In recent years, the field of mechanistic interpretability has emerged to address this challenge. Mechanistic interpretability aims to understand the inner workings of neural networks by investigating their parameters and activations. The ideal goal is to reverse-engineer the algorithm that a neural network implements into a fully human-interpretable form, such as a computer program.

Fundamental notions commonly used in mechanistic interpretability are *features*, *circuits*, and *representations*. However, the notion of a feature does not have a universally accepted, computationally meaningful definition. In general contexts, a *feature* is a property of the input data that is relevant for the task at hand. In mechanistic interpretability specifically (e.g., in [3]), a *feature* is more concretely understood as an interpretable concept that is computed by a neural network. Examples of identified features include curve detectors of specific orientations in [4] (activating for curved line-like shapes with a fixed orientation), high-low frequency detectors in vision models in [3] (detecting a boundary between a high-frequency and low-frequency region in an image), and verbal concepts such as the Golden Gate Bridge in [5] (activating for expressions related to the Golden Gate Bridge and the word itself).

Building on this, a *representation* is a mapping from input data to a feature space, encoding these concepts in the network's activations. A *circuit* is a computational substructure within a neural network that computes a specific feature.

Some clearly human-interpretable concepts like a specific curve detector or a 'cat' feature often appear to be represented. However, individual neurons are often polysemantic, meaning that they compute multiple features at once. For example, in [3] a neuron is discussed that activates for semantically unrelated concepts such as cat faces, fronts of cars, and legs. Similarly, [6] documents neurons in the vision model CLIP that respond to disparate concepts such as "turtle" and "PhD" or "dice" and "poet". These concepts appear semantically unrelated and uncorrelated in typical inputs. This suggests the neuron is not detecting a feature common to these inputs, but rather responding to multiple distinct features simultaneously.

One explanation for polysemantic neurons is superposition: representing more features than there are available dimensions. This is possible when features are *sparse*, meaning they are inactive in the vast majority of inputs. In such cases, multiple feature directions can share the same physical dimensions with minimal interference, as they rarely activate simultaneously. Superposition allows for both representation and computation of significantly more features than dimensions in the feature space. This is hypothesized to be a key factor in the efficiency of transformer architectures.

For example, in transformers, the residual stream dimension is often much smaller than the total number of features the model appears to compute with, which suggests use of superposition.[1] [2]

Superposition presents significant challenges to decomposing neural networks into modular algorithms. Finding an overcomplete set of feature directions requires solving an underdetermined inverse problem. Moreover, verifying that the discovered set of features is complete is fundamentally difficult, as there is mostly no ground truth to compare against. Superposition has also been linked to facilitating adversarial vulnerability [1] and overfitting [8]. Eliminating superposition is in principle possible, e.g., via l_1-regularization or modified activation functions. However, this also appears to reduce the performance of the model (e.g., in [9], with a modified activation function). This performance reduction likely occurs by disabling the advantages of superposition.

Given the hypothesis that superposition is a key mechanism in trained neural network representations, decomposing parameters into interpretable feature vectors becomes a critical problem. Sparse dictionary learning (or sparse autoencoders, SAE) has emerged as a common approach toward this goal. In this approach, one inserts a large bottleneck layer into the neural network between the layers of interest. The bottleneck layer is then trained with a sparsity penalty while fixing the rest of the network parameters. Mostly, the vectors of the bottleneck layers are interpretable and monosemantic, see [10]. One can verify a causal effect on the model's behavior by modifying the activations of these bottleneck neurons in the forward pass of the model, such as in [5].

However, sparse autoencoders have several limitations. These include absorbed or split features, as well as inability to decompose circuits. The sparsity constraint only encourages individual features to be sparse, but it does not ensure an independent feature decomposition. For example, multiple features co-activating frequently might be merged into a single feature in the bottleneck layer. Alternative methods for network parameter decomposition have been proposed, such as attribution-based parameter decomposition [11], which aims to directly decompose the full parameter set into sparse components based on causal attribution.

1.1 Overview

This thesis is structured as follows:

- In this chapter (Chapter 1), we provide an introduction to the main concepts of mechanistic interpretability, representations, and superposition in neural networks, as well as an overview of related work.
- In Chapter 2, we give a synthesis of fundamental definitions and statements for understanding superposition.
- In Chapter 3, we examine empirical results on representational superposition, following the influential article "Toy Models of Superposition" [1]. We provide precise mathematical statements for the main theoretical results and investigate the emergence of superposition under different conditions. We also investigate the polygon solutions seen in the original article in greater detail, finding results on their stability and behavior.
- In Chapter 4, we investigate random models of Boolean computation in superposition. We provide detailed proofs for constructions from the main part of [2], for which only proof sketches were

[1] The residual stream is the shared activation space of all layers in a transformer model. Note that in a residual network such as a transformer, each layer's output is added to its input. This implies that all layers operate on a common vector space, the residual stream. Consequently, the residual stream dimension d_{model} of a transformer model is the dimension of the vectors that are passed between transformer blocks.

[2] As an estimate, the authors in [5] use feature counts for autoencoders in the range of $[1, 34] \cdot 10^6$ features for their Claude Sonnet 3 model, which is much larger than the residual stream dimensions. The authors do not publish the residual stream dimension of Claude Sonnet 3. However, we can estimate that it is likely within $d_{\text{model}} \in [5, 100] \cdot 10^3$. For example, in GPT-3, the residual stream dimension is $d_{\text{model}} = 12288$ [7].

given in the original article. We extend the results to proper exponential scaling of the width. By extending the constructions, we find slightly weaker bounds for the scaling using their Bernoulli construction. We also provide approximate bounds for several of the constructions and validate them empirically.
- In Chapter 5, we summarize the main results from this thesis and discuss open problems and directions for future research.

1.2 Related Work

We give an overview of related work in the fields of mechanistic interpretability, representations in neural networks, superposition, and sparse autoencoders.

1.2.1 Interpretability and Features

There is extensive literature on feature visualization in neural networks. For example, early work such as [12] and [13] show visualization of features in vision models.

The hypothesis that there is interpretable decomposable structure in neural networks has been explored in [14] and [3].

Notable counter-evidence against general interpretability and universality exists in the literature. For example, [15] finds that CNNs tend to learn surface statistical regularities rather than high-level concepts by training networks on different Fourier-filtered versions of one dataset. Similarly, [16] demonstrates that adversarial examples can be constructed from robust and non-robust features, where non-robust features are imperceptible to humans but predictive for the model. Also, [17] finds that ImageNet-trained vision models often rely on texture instead of shape, which differs from human perception, indicating that models generally learn different features than humans do.

Other foundations of mechanistic interpretability have also been proposed, such as viewing neural networks as approximately piecewise linear functions and interpreting them via their linear regions, as in [18].

Interpretability of circuits and features in large language models has been investigated in multiple works. For example, [19] provides evidence for an "indirect object identification" circuit in GPT-2 Small, which enables simple grammatical completion. For instance, the phrase "When Mary and John went to the store, John gave a drink to" can be completed by "Mary" via this heuristic.

1.2.2 Representations

Studying representations is a fundamental aspect of understanding internals of neural networks and has been investigated in multiple works.

A well-known result on neural network representations is that they are *hierarchical* in nature. For example, in [13, p. 7] it is found that in an ImageNet model, lower layers represent low-level features such as edges and colors, while higher layers represent more complex features such as object parts and entire objects. This gives a natural explanation for the generalization advantage and efficiency of deep networks compared to shallow networks.

Models of distributed representations in neural networks have been hypothesized and studied for several decades. Similar problems have been studied in neuroscience, where it has been hypothesized that the brain represents information in a distributed manner across populations of neurons (e.g., [20]). For example, in [21] a way of learning distributed representations is proposed.

1.2.3 Linear Representations

Since neural networks are built from linear operations such as matrix multiplications and vector additions, it is natural to study linear structure in neural network representations. Notably, residual

networks and transformers exhibit even stronger shared linear structure (see e.g. [22]): Outputs of subsequent layers are added to the input. This encourages a joint representation across layers in a common vector space, which is called the residual stream.[1]

In the well-known paper [23], the authors train a recurrent neural network on word prediction and use the hidden state of the network as a vector representation of the word. They find that the hidden state of the network captures semantic relationships between words in an affine way, such as $V(\text{queen}) - V(\text{king}) \approx V(\text{woman}) - V(\text{man})$, where V denotes the vector representation of a word. Similar results have been found in other models, such as Generative Adversarial Networks (GANs) [24], where the authors find that vector arithmetic in the latent space of the GAN corresponds to semantic changes in the generated images, such as removing windows from images.

There are important distinctions in different concepts of linear representation. Testing if a concept is linearly represented is often done via a linear probe, i.e., training a linear classifier (read-off vector) on top of the activations to predict the presence of a concept [25]. However, this does not necessarily imply that the read-off vector is a suitable steering vector, i.e., that adding the read-off vector to a representation causes downstream output to change reliably according to the concept. This distinction is detailed in [26], where the authors subdivide between "linear measurability" and "intervention-based" *linear representation* in transformers.

1.2.4 Superposition

The general concept of sparse coding is well studied in both mathematics and neuroscience. However, the specific hypothesis that superposition might be a key mechanism in modern neural networks has been explored only more recently.

Notably, in [14] superposition is discussed as a possible explanation for polysemantic neurons in neural networks.

In [27], the authors provide one of the first empirical demonstrations of superposition. They give evidence that word embeddings produced by polysemantic words are often a linear superposition of the individual meanings of that word. For example, the word embedding for "tie" could be decomposed into separate components corresponding to "necktie" and "tie" in the sense of a draw in sports as $v_{\text{tie}} \approx \alpha_1 v_{\text{tie}_1} + \cdots + \alpha_n v_{\text{tie}_n}$,where v_{tie_i} is the vector representation of the i-th meaning of "tie" and $\{\alpha_i\}_i$ are weighting factors.

In the popular blog article [1] from Anthropic, the authors investigate a toy model for the general phenomenon of superposition, which we base our investigation of representational superposition (Chapter 3) on. This work first showed an empirical model of toy networks learning to represent features in superposition. For example, it demonstrated that a two-dimensional neural network layer will often learn to represent five features in superposition when features are sufficiently sparse, organizing them in a pentagon configuration.

In the follow-up article [8], the authors investigate the same model in an overfitting setting. They find that superposition is a natural way for neural networks to memorize random data efficiently and links the double-descent phenomenon to different representation regimes.

Leveraging superposition for computation has been investigated in [2], where the authors give random constructions for Boolean computation in superposition. One of the fundamental results is that computation in superposition allows only a polynomial number of features per dimension, as opposed to an exponential number of features per dimension for representation in superposition. Based on similar complexity techniques, the unpublished preprint [28] gives general upper and lower bounds for exact Boolean computation in superposition.

Computation in superposition has been extended to arbitrary computations in the Alignment Forum article [29] (follow-up correction in [30]). Here, the authors demonstrate composition of arbitrary circuits computed by smaller sub-networks into a larger network computing all sub-circuits in superposition. They find that the maximum number $T_{\max}$ of d-dimensional circuits that can be composed in a large network of size D scales as $T_{\max} = \mathcal{O}\left(\frac{1}{z}\frac{D^2}{d^2}\right)$, where z is the maximum number of circuits active in a forward pass.

1.2.5 Identifying Features in Superposition

Since the introduction of the concept of superposition in neural networks, identifying the feature vectors that are represented in the parameters has been a question of significant interest.

Sparse autoencoders for feature extraction in the context of feature superposition were introduced in [31] and investigated in greater detail in [10].

Sparse autoencoders have been scaled to progressively larger models, such as in [32] and [5]. While it is not clear whether the vectors extracted by the autoencoders correspond to the features that the model uses for computation, they have been used to steer models towards specific behaviors. For example, in [5], the authors identify and manipulate features corresponding to concepts such as the Golden Gate Bridge, inner conflicts, and deceptive behavior. They demonstrate using feature vectors for steering, such as writing scam emails without refusing when a 'scam email' feature is clamped to a high value.

Sparse autoencoders have several limitations, such as appropriately choosing the number of features (the size of the bottleneck layer) for a given model. Another fundamental problem is 'feature absorption', demonstrated in [33], where one autoencoder latent seemingly corresponds to a meaningful, monosemantic feature, but the latent does not activate in some inputs where it naturally should, and instead, a different latent activates, 'absorbing' the feature. Also, in [34], the authors find that some features involving modular addition, such as days of the week, are organized with their relevant individual features in a meaningful circular structure in a not-almost-orthogonal way. This suggests that models might learn more complex geometric organizations of features than unstructured almost-orthogonal directions for computation purposes.

Notably, in [11] (with follow-up [35]), the authors propose an alternative method for directly linearly decomposing the full parameters of a neural network into sparse components based on causal attribution and minimizing a proxy for description length of the individual components. While the authors find that this method addresses shrinkage and absorption issues of sparse autoencoders, it is computationally expensive.

Chapter 2

Features and Superposition

Empirical considerations, such as the number of learned distinct concepts in LLMs, suggest that neural networks represent more distinct concepts than the dimensionality of their activation spaces would naively allow.

The *superposition hypothesis* [1] proposes that networks achieve this by representing features as nearly-orthogonal directions in activation space, trading perfect orthogonality for increased representational capacity. Rather than aligning each feature with a basis vector (which would limit representation to d features), networks use an overcomplete set of directions, tolerating small interference between non-orthogonal features.

Our goal in this chapter is to provide definitions of features and superposition in neural networks and establish the elementary theoretical estimates for the problem of representational superposition —how many features can be represented in a layer, and under what conditions. This chapter addresses three core questions:

(i) How many features can theoretically be represented in a layer, and how does this number scale with dimensionality?
(ii) What does it mean for a feature to be represented, and when is approximate representation sufficient?
(iii) How can we characterize the independence structure of features?

2.1 Setup and Foundational Definitions

In this section, we establish foundational definitions and notation used throughout this chapter and subsequent chapters. An overview of notation can be found in Appendix A.1.

Note 2.1. (Setup):
We consider a trained network with function $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} := \mathbb{R}^{d_x}$ is the input space and $\mathcal{Y} := \mathbb{R}^{d_y}$ is the output space.

When considering probabilistic notions, we assume a probability measure $\mathbb{P}_{\mathcal{X}}$ with an associated σ-algebra of events $\mathcal{G}_{\mathcal{X}}$ to be given and consider the input to be a random variable X. We write $\mathcal{A} \in \mathcal{G}_{\mathcal{X}}$ for the corresponding events. Usually we omit mentioning the explicit σ-algebra of the space. When the attribution is clear and there is no ambiguity, we write $\mathbb{P}(X \in \mathcal{A})$ or $\mathbb{P}(X = x)$ for the respective probabilities.

When discussing feature representations in a layer, we denote the activations of the l —th layer $\text{act}^l : \mathcal{X} \to \mathbb{R}^{d_l}$. When considering a general layer, we also may drop the index l and write $\text{act} : \mathcal{X} \to \mathbb{R}^{d_l}$. We also call the number of dimensions in the activation space d_l the *width* of the layer.

Example 2.2. (Simple feed-forward network):
As an example, consider a simple ReLU-feed-forward network for classifying MNIST digits with $d_{\mathcal{X}} = 28 \times 28$ and $d_{\mathcal{Y}} = 10$ with 3 layers. Here the activations are computed as

$$\begin{aligned} \mathrm{act}^0(x) &= x \\ \mathrm{act}^l(x) &= \mathrm{ReLU}\big(\boldsymbol{W}^l \mathrm{act}^{l-1}(x) + b^l\big) \qquad \forall l \in 1{:}2, \\ \mathcal{M}(x) &= \mathrm{act}^3(x) = \mathrm{softmax}(\boldsymbol{W}^3 \mathrm{act}^2(x) + b^3) \in \mathcal{Y}, \end{aligned}$$

with weight matrices $\boldsymbol{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}$ and biases $b^l \in \mathbb{R}^{d_l}$, with $\mathrm{ReLU}(x) \coloneqq \max(0, x)$ and the softmax operation defined as $\mathrm{softmax}(z)_i \coloneqq \frac{\exp(z_i)}{\sum_j \exp(z_j)}$ for $z \in \mathbb{R}^{d_y}$.

Whenever applying a one-dimensional function in a network context to multi-dimensional inputs, we usually mean the function is applied component-wise, i.e., $\mathrm{ReLU}(x) = (\mathrm{ReLU}(x_1), \mathrm{ReLU}(x_2), ..., \mathrm{ReLU}(x_n))$ for $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$.

2.2 Features and Linear Representations

To formalize the superposition hypothesis, we require precise definitions of what constitutes a "feature" and what it means for a feature to be "represented" in a layer's activations. Historically, the notion of a feature lacks a universally agreed-upon definition—in [1], for instance, three distinct characterizations are given. These difficulties mainly arise, because it is not clear what fundamental unit to decompose a network into is or whether this is meaningfully possible at all.

In this thesis, we formally understand a feature as a quantity that could be computed from the input by a neural network. This definition is deliberately general and, crucially, does not presuppose that the feature is actually represented in any particular layer, which we define separately. This generality allows us to discuss both features that are represented and features that could be represented but are not.

The notion of a feature is fundamental in *mechanistic interpretability*, which studies internal representations with the goal of understanding neural network behavior at a fine-grained level.

Definition 2.3. (Feature, active feature):
A *feature* is an $\mathbb{R}^{d_f}$-valued function of the network input, i.e., a function $f : \mathcal{X} \to \mathbb{R}^{d_f}$ for some dimension $d_f \in \mathbb{N}$. We say that a feature is *active* for an input or set of inputs if it is nonzero for that input or that set of inputs. We write $F = f(X)$ for the corresponding random variable.

In order to discuss which features appear to be computed in a network, we need to define a notion of a feature being represented (i.e., being computed) in the activations of a layer. Because weight matrices operate on the activations of a layer in a linear fashion, it seems natural that at least some features might combine linearly.

Definition 2.4. (Linearly represented feature):
A one-dimensional feature $f : \mathcal{X} \to \mathbb{R}$ is *linearly represented* in a layer with activations $\mathrm{act} : \mathcal{X} \to \mathbb{R}^{d_l}$, if it can be linearly recovered by a *read-off vector* $r \in \mathbb{R}^{d_l}$, i.e.,

$$f(x) = \mathrm{act}(x) \cdot r \qquad \forall x \in \mathcal{X}.$$

Note, in particular, that a feature can be represented (i.e., can be read-off from the activations), but not actually be used in that precise form in the network (for example, changing the activations by subtracting the corresponding directions does not have a downstream causal influence).

If all features are linearly represented, then the network activations can be fully represented as a linear combination of an orthonormal basis of feature directions r, which function as their own features' read-offs, i.e.,

$$\mathrm{act}(x) = \sum_k f_k(x)\hat{r}_k \qquad \forall x \in \mathcal{X}. \tag{2.1}$$

In this case, a network could only represent d_l features and their linear combinations, which could be recovered by orthogonal decomposition. The network would not lose representational power by aligning features with the activation space axes, providing no explanation for the empirical observation of *polysemantic features*—individual neurons activating for semantically unrelated inputs.

The superposition hypothesis resolves this by proposing that networks represent more than d_l features by leveraging *almost-orthogonal* directions. The feature directions form an overcomplete spanning set (an *overcomplete basis*), sacrificing perfect orthogonality for increased capacity. However, non-orthogonality introduces interference: reading off a feature f inevitably reads off small components of other non-orthogonal features. This motivates a notion of approximate linear representation that tolerates bounded error.

Definition 2.5. (One-dimensional ε-linearly represented feature):
We call a feature $f : \mathcal{X} \to \mathbb{R}$ ε-linearly represented in a layer with activations $\mathrm{act} : \mathcal{X} \to \mathbb{R}^{d_l}$, if it can be linearly recovered by a *read-off vector* $r \in \mathbb{R}^{d_l}$ with maximum error ε, i.e.,

$$|\mathrm{act}(x) \cdot r - f(x)| < \varepsilon \qquad \forall x \in \mathcal{X}.$$

Notably, this definition is not scaling-invariant, but rather equivariant, i.e., if a feature f is ε-linearly represented with read-off vector $r \in \mathbb{R}^{d_l}$, then αf (for $\alpha > 0$) is $\alpha\varepsilon$-linearly represented with the scaled read-off vector αr. (I.e., if one considers a feature to have an associated unit, then the error ε has the same unit as f)

The definition of ε-linear representation contains a subtle but critical constraint: interference from non-orthogonal features compounds when multiple features are simultaneously active. If all $m > d_l$ features could be active simultaneously, the ε error from each non-orthogonal direction would accumulate, potentially making accurate read-off impossible regardless of how small ε is chosen. The definition is therefore only meaningful when features exhibit *sparsity*—at most $k \ll m$ features are active for any given input.

Consider an overcomplete set of feature vectors $\phi_1, \ldots, \phi_m$ with $m > d_l$. Because the first d_l vectors span the space, if all d_l corresponding features are simultaneously active, reading off the remaining $m - d_l$ features may yield arbitrary values even when those features are inactive. Thus, counterintuitively, whether a feature is ε-linearly represented depends not only on the geometry of read-off vectors but also on the input distribution—specifically, on which combinations of features can co-occur.

Definition 2.6. (Sparsity):
We say that a set of features $f_1, ..., f_n$ in a layer is k-sparse if at most k features are active for a given input, i.e. the number of non-zero elements of the vector $f = (f_1, ..., f_n)$ is at most k, i.e., $\|f\|_0 \leq k$. We say that features are sparser if the number of active features (i.e., sparsity) is lower.

Having established the definitions of features, linear representation, and sparsity, we can now address the core questions motivating this chapter:

(i) How many features can be represented in a d_l-dimensional layer, and how does this capacity scale with d_l?
(ii) How can we identify which features are represented in a layer?
(iii) How can a network perform computations on superposed representations?

We refer to the problem of storing and retrieving features from activations as *representational superposition* and the problem of performing operations on superposed features as *computational superposition.*

The distinction is fundamental: representational superposition only requires that features can be encoded in and decoded from activations (via linear read-off), whereas computational superposition requires that subsequent layers can directly operate on these overlapping representations. This chapter primarily addresses representational superposition, establishing theoretical bounds on capacity. Computational superposition is analyzed in Chapter 4.

2.3 Basic Theoretical Bounds

We now address the first question: how many features can theoretically coexist in superposition? The answer hinges on the geometric constraint that feature directions must be nearly orthogonal to permit low-interference read-off. The *Johnson-Lindenstrauss lemma* provides a fundamental existence guarantee for the number of approximately orthogonal vectors in a given dimensionality.

To formalize the notion of "nearly orthogonal," we give the notion of ε-orthogonality.

Definition 2.7. (ε-orthogonality [34, analogous to Def. 4]):
We call two vectors $u, v \in \mathbb{R}^d \setminus \{0\}$ ε-orthogonal if the absolute value of their normalized inner product is less than ε, i.e.,

$$|\hat{u} \cdot \hat{v}| < \varepsilon,$$

where $\hat{u} = \frac{u}{\|u\|_2}$ and $\hat{v} = \frac{v}{\|v\|_2}$ are the normalized vectors.

We call two matrices $\boldsymbol{A} \in \mathbb{R}^{d \times d_1}, \boldsymbol{B} \in \mathbb{R}^{d \times d_2}$ ε-orthogonal if any two unit vectors from the column spaces of $\boldsymbol{A}$ and $\boldsymbol{B}$ are ε-orthogonal, i.e.,

$$|\hat{u} \cdot \hat{v}| < \varepsilon$$

for all unit vectors $\hat{u} \in \text{colspace}(\boldsymbol{A}), \hat{v} \in \text{colspace}(\boldsymbol{B})$.

For representational superposition, a corollary of the *Johnson-Lindenstrauss lemma* gives us a guarantee of the number of ε-orthogonal vectors that can appear in a d-dimensional space.

The Johnson-Lindenstrauss-Lemma generally states that a set of points in a high-dimensional space can be linearly embedded into a lower-dimensional space while preserving the pairwise distances up to a factor of $1 \pm \varepsilon$. Here, the vectors are usually constructed using random projections. There are several variants of the Johnson-Lindenstrauss lemma, where many only give Landau bounds. We use a version from [36], which gives a precise bound on the distortion factor ε.

Theorem 2.8. (Johnson-Lindenstrauss lemma, [36, Thm 2.1]):
Let $0 < \varepsilon < 1$ and let a set V of n points in a space of any dimension $\mathbb{R}^d$ be given. Let $k \in \mathbb{N}$ such that

$$k \geq 4\log(n)\left(\frac{1}{2}\varepsilon^2 + \frac{1}{3}\varepsilon^3\right)^{-1}. \tag{2.2}$$

Then there exists a linear mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ such that the squared Euclidean distances between points in V are preserved up to a factor of $1 \pm \varepsilon$, i.e.,

$$(1-\varepsilon)\|u-v\|_2^2 \leq \|f(u)-f(v)\|_2^2 \leq (1+\varepsilon)\|u-v\|_2^2 \qquad \forall u, v \in V.$$

Corollary 2.9. (JL-lemma for scalar products):
Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$, $k \in \mathbb{N}$ be such that

$$k \geq 4\log(n+1)\left(\frac{1}{2}\varepsilon^2 + \frac{1}{3}\varepsilon^3\right)^{-1} \in \Theta(\log(n)\varepsilon^{-2}).$$

Then there exists a set of n vectors in $\mathbb{R}^k$ with norm $\in [1 \pm \varepsilon]$ and maximal absolute scalar product 2ε of distinct vectors. That is, there exist $\{v_i\}_{i=1}^n \subset \mathbb{R}^k$ with

$$\left|v_i \cdot v_j\right| \leq 2\varepsilon \qquad \forall i, j \in 1{:}n, i \neq j.$$

Proof.
Choose $V := \{0, e_1, ..., e_n\}$ to be the set containing the zero vector and the standard basis vectors in $\mathbb{R}^d$ with $d = n$. Then, the Johnson-Lindenstrauss-Lemma yields vectors $v_0 = f(0) = 0$, $v_1 = f(e_1)$,..., $v_n = f(e_n)$.

From the JL-distance preservation, due to linearity of f we directly obtain that the squared norm of a vector is preserved up to $1 \pm \varepsilon$, i.e.

$$(1-\varepsilon)\|e_i - 0\|_2^2 \leq \|v_i - f(0)\|_2^2 \leq (1+\varepsilon)\|e_i - 0\|_2^2,$$

from which the norm boundedness directly follows, i.e., $1 - \varepsilon \leq \|v_i\|_2^2 \leq 1 + \varepsilon$.

Now, to obtain a bound on the scalar product, we use the polarization identity $u \cdot v = \frac{1}{2}(\|u\|_2^2 + \|v\|_2^2 - \|u-v\|_2^2)$. We consider distinct vectors v_i, v_j with $i \neq j$. Employing the polarization identity for both $u = e_i, v = e_j$ and $u = v_i, v = v_j$, we obtain

$$\begin{aligned}
&\left|v_i \cdot v_j\right| = \left|v_i \cdot v_j - e_i \cdot e_j\right| && \downarrow \text{Term to bound} \\
&= \frac{1}{2}\left|\left(\|v_i\|_2^2 + \|v_j\|_2^2 - \|v_i - v_j\|_2^2\right) - \left(\|e_i\|_2^2 + \|e_j\|_2^2 - \|e_i - e_j\|_2^2\right)\right| && \downarrow \text{Polarization Identity} \\
&\leq \frac{1}{2}\left[\left|\|v_i\|_2^2 - \|e_i\|_2^2\right| + \left|\|v_j\|_2^2 - \|e_j\|_2^2\right| + \underbrace{\left|\|e_i - e_j\|_2^2 - \|v_i - v_j\|_2^2\right|}_{\leq 2\varepsilon \text{ (2.3.1)}}\right] && \downarrow \text{Rearrange, bound} \\
&\leq \frac{1}{2}(\varepsilon + \varepsilon + 2\varepsilon) = 2\varepsilon,
\end{aligned}$$

which gives the final bound on the absolute scalar product of distinct vectors.

For the bounding of the rightmost term, we used a rearrangement of the JL-inequality: Starting from the JL-inequality for our vectors, we have

$$\begin{aligned}
&(1-\varepsilon)\|e_i - e_j\|_2^2 \leq \|v_i - v_j\|_2^2 && \leq (1+\varepsilon)\|e_i - e_j\|_2^2 \\
\Rightarrow \quad &\underbrace{-\varepsilon\|e_i - e_j\|_2^2}_{2} \leq \|v_i - v_j\|_2^2 - \|e_i - e_j\|_2^2 && \leq \underbrace{\varepsilon\|e_i - e_j\|_2^2}_{2} \\
\Rightarrow \quad &\left|\|v_i - v_j\|_2^2 - \|e_i - e_j\|_2^2\right| \leq 2\varepsilon. && (2.3.1)
\end{aligned}$$

□

Remark 2.10. (Principal estimates for representational superposition):
To obtain n vectors via Corollary 2.9 with scalar product at most 2ε in k dimensions, we need

$$k \geq 4\log(n+1)\left(\frac{1}{2}\varepsilon^2 + \frac{1}{3}\varepsilon^3\right)^{-1}.$$

Rearranging gives

$$n \leq \exp\left(\frac{1}{4}k\left[\frac{1}{2}\varepsilon^2 + \frac{1}{3}\varepsilon^3\right]\right) - 1,$$

i.e., the number of features n that can in principle be represented in a layer with k dimensions with approximately ε'-orthogonal directions rises exponentially with the number of dimensions k. Note that this ignores practical considerations such as space constraints in floating point representation.

2.4 Reducibility and Independence of Features

The Johnson-Lindenstrauss bound establishes that superposition can in principle exponentially expand representational capacity: a k-dimensional layer can represent $\exp(\mathcal{O}(k))$ nearly-orthogonal features. However, we have so far assumed that features are one-dimensional and approximately orthogonal, while at least some features of interest (e.g., modular quantities, days of the week) seem to be represented in a multi-dimensional way [34].

Given the empirical evidence that concepts can often be recovered by linear probes (e.g., [25]), one can ask whether features can always be linearly decomposed into independent features and how to rigorously define when two features are "distinct" or "independent".

In the following section, we discuss the notion of reducibility of features, which was introduced in [34]; however, the definition of reducibility given is not formally consistent, which is why we use an adapted definition here. We discuss the original version in Section 2.5.

Definition 2.11. (Reducibility):
A feature $F : \mathcal{X} \to \mathbb{R}^d$ is reducible into features $G : \mathcal{X} \to \mathbb{R}^k$, $H : \mathcal{X} \to \mathbb{R}^{d-k}$, with $k, d-k > 0$, if there is an affine transformation with an orthonormal matrix $\boldsymbol{R} \in \mathbb{R}^{d \times d}$ and a vector $v \in \mathbb{R}^d$ such that

$$\begin{pmatrix} G \\ H \end{pmatrix} = \boldsymbol{R}F + v,$$

where for G, H it holds any of the following:

(i) (independence): G and H are stochastically independent, i.e., $\mathbb{P}(G \in \mathcal{A}_G, H \in \mathcal{A}_H) = \mathbb{P}(G \in \mathcal{A}_G)\ \mathbb{P}(H \in \mathcal{A}_H)$, or

(ii) (mixture of mutually exclusive): the joint distribution of G and H is such that the probability of G and H being active for the same input is 0, i.e., $\mathbb{P}(G \neq 0, H \neq 0) = 0$.

We say that a feature is separable if it is reducible into stochastically independent features, and a feature is a mixture, respectively, if it is reducible into a mixture of (mutually exclusive) features.

Note 2.12. (Equivalent notions of a feature being a mixture):
A feature F is a mixture of features G, H iff:

(i) the probability of G and H being active for the same input is 0, i.e., $\mathbb{P}(G \neq 0, H \neq 0) = 0$.

(ii) the joint distribution can be expressed as a mixture of mutually exclusive distributions, such that

$$\begin{aligned} \mathbb{P}(G \in \mathcal{A}_G, H \in \mathcal{A}_H) = p_G\ &\mathbb{P}(G \in \mathcal{A}_G \mid G \neq 0)\delta_0(\mathcal{A}_H) \\ + p_H\ &\mathbb{P}(H \in \mathcal{A}_H \mid H \neq 0)\delta_0(\mathcal{A}_G) \\ + p_0 &\delta_0(\mathcal{A}_G)\delta_0(\mathcal{A}_H) \end{aligned}$$

with $p_G = \mathbb{P}(G \neq 0)$, $p_H = \mathbb{P}(H \neq 0)$, $p_0 = \mathbb{P}(G = 0, H = 0)$ and δ being the Dirac measure, i.e., $\delta_x(\mathcal{A}) = 1(x \in \mathcal{A})$.

Example 2.13. (Examples of reducible and irreducible features):
The reducibility covers two distinct cases:

- The first case of reducibility into stochastically independent features, such as the classical toy features used in [1]:

$$F_i \sim \mathcal{U}[0,1] \qquad \text{i.i.d. for } \ i \in 1{:}m.$$

- The mixture case covers, e.g., one-hot encoded features, such as tokens F:

$$F = e_I \qquad \text{where } \ I \text{ is uniformly distributed in } \ 1{:}m.$$

Note 2.14. (Geometric implications of reducibility in 2D):
Let $\tilde{F} \in \mathbb{R}^2$ be a feature with finite support (i.e., a discrete random variable which takes on a finite number of values). We understand $\tilde{F}$ here as the transformed version of the original feature F in Definition 2.11. Let $G := \tilde{F} \cdot e_1$, $H := \tilde{F} \cdot e_2$ as in Definition 2.11. Then, if G, H are stochastically independent, then the support of $\tilde{F}$ is a cartesian product of the supports of G and H, i.e.,

$$\operatorname{supp}\left(\tilde{F}\right) = \operatorname{supp}(G) \times \operatorname{supp}(H).$$

If G, H form a feature mixture, then

$$\operatorname{supp}\left(\tilde{F}\right) = \operatorname{supp}(G) \times \{0\} \cup \{0\} \times \operatorname{supp}(H),$$

i.e. the support of $\tilde{F}$ is a subset of the axes $\mathbb{R} \times \{0\} \cup \{0\} \times \mathbb{R}$.

We now illustrate the definition of reducibility with a common example of irreducible features, circular features.

Example 2.15. (Irreducibility of discrete circular features):
Define the 2-dimensional regular polygon $C_n := \left\{\left(\cos 2\frac{\pi}{n}i, \sin 2\frac{\pi}{n}i\right) \mid i \in 0{:}n-1\right\} \subset \mathbb{R}^2$.
Define the "polygonal feature" F_n to be uniformly distributed over C_n.

Then, for $n \geq 5$ the polygonal feature F_n is irreducible, because there is no way to rotate the regular polygon C_n to either align with both axes or such that it is representable as a cartesian product of two sets.

For $n \leq 4$, F_n can be reduced:

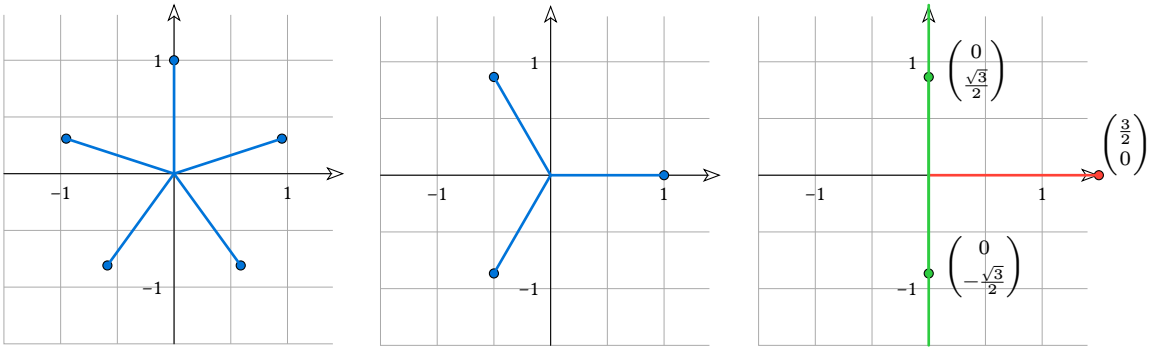
(i) For $n = 1$, the resulting "point" feature can be reduced into two independent features, one completely inactive feature and one feature that is active for all inputs: Let $G := F_1 \cdot e_1 \equiv 1, H := F_1 \cdot e_2 \equiv 0$ (Choose the identity rotation and zero translation). Then G, H are constant, thus stochastically independent, both are trivially irreducible because 1 cannot be subdivided into a sum of positive integers.

(ii) For $n = 2$, the resulting "line" feature can be reduced, e.g. into two separable features (With another rotation, one could also decompose it as a mixture). Let $G := F_2 \cdot e_1, H := F_2 \cdot e_2 \equiv 0$. Then G, H are stochastically independent, because H is constant, thus are separable features.

(iii) For $n = 3$, the resulting "triangle" feature can be reduced into a mixture of one completely active and one "up-or-down" feature. The idea is to shift the triangle, such that the left side aligns with the second axis: $C_3 + \left(\frac{1}{2}, 0\right) = \left\{\left(\frac{3}{2}, 0\right), \left(0, \frac{\sqrt{3}}{2}\right), \left(0, -\frac{\sqrt{3}}{2}\right)\right\} \subset \mathbb{R} \times \{0\} \cup \{0\} \times \mathbb{R}$.

(iv) For $n = 4$, the resulting "square" feature is reducible, either into two separable features or into a mixture of features, depending on the rotation chosen (identity or $45°$ rotation).

In particular, we see that the decomposition is not generally unique (see Figure 1).



Pentagon Feature | Original 'Triangle' Feature | Translated Triangle Feature

Figure 1: An irreducible pentagon feature F_5 (left) and a reducible triangle feature F_3 (right). The triangle feature can be reduced into a mixture of two features, one completely active and one "up-or-down" feature. The regular n-gon feature is irreducible for $n \geq 5$, because there is no way to rotate the regular polygon C_n to either align with both axes or such that it is representable as a cartesian product of two sets.

2.5 Alternative Definitions of Reducibility and Features

In this section, we discuss the definitions of features and reducibility introduced in [34] and point out arising issues.

For this section, we use the original notation from [34], i.e., f, a, b are both considered random variables and functions, depending on the context.

Note 2.16. (Definition of feature in [34]):
A feature is a function from a subset of the input space to $\mathbb{R}^d$. The feature is said to be active on the aforementioned subset [and inactive otherwise].

Note 2.17. (Original definition of reducibility [34, Def. 2]):
A feature f is reducible into features a and b if there is an affine transformation

$$\boldsymbol{R}F + v = \begin{pmatrix} a \\ b \end{pmatrix},$$

with R orthonormal, and $a \in \mathbb{R}^d$, $b \in \mathbb{R}^{d-d_G}$ considered as random variables, such that

(i) p is *separable*, i.e., $p(a, b) = p(a)p(b)$ or

(ii) p is a *mixture*, i.e., here a sum of disjoint distributions, one of which is lower-dimensional, i.e.,

$$p(a, b) = wp(a)\delta(b) + (1 - w)p(a, b)$$

with δ being the Dirac delta "function" and $0 < w < 1$. Here p is a probability density function conditional on the subset of the input space where f is active.

Note 2.18.
In [34], the authors claim that p being a mixture in this sense is equivalent to:

(i) The joint distribution p is a sum of disjoint distributions.
(ii) The features a, b are not simultaneously being active.
(iii) There is no set where both have positive probability measure.

Notably, by their definition of a feature being active, both features a, b are trivially active for every input for which f is active, so condition (ii) is never true by the literal definition of a feature being active.

Using $\delta(b)$, it is likely meant that b should be considered inactive if it is zero. Also, for their "updated superposition hypothesis" [34, Hypothesis 1], they extend a feature to be 0 on all inputs where it is not active, which means that an active feature with value 0 cannot be distinguished from an inactive feature in any network representation.

2.6 Summary

In this chapter, we established the foundational framework for understanding superposition in neural networks. We used the formalization of features as functions of network inputs and defined what it means for features to be linearly represented in a layer's activations. The key insight is that ε -linear representation—tolerating small read-off errors—in principle enables networks to represent far more features than dimensions, provided features exhibit sparsity.

The Johnson-Lindenstrauss lemma provides a constructive theoretical estimate: exponentially many nearly-orthogonal directions can coexist in a fixed-dimensional space, suggesting that k-dimensional layers can represent $\exp(\mathcal{O}(k))$ features. However, this bound applies to representational superposition and does not address computationally performing operations on superposed representations.

Finally, we examined the structure of individual features through the lens of reducibility (stochastically independent components) and mixtures (mutually exclusive components) from [34]. We illustrated these concepts with examples and discussed the relation of the original reducibility definition in [34] and our adapted definition.

Chapter 3

Representational Superposition

In Chapter 2, we established the theoretical possibility of representational superposition in neural networks. However, from this, it is not clear how robustly and under which conditions superposition actually emerges in practice.

In this chapter, we investigate a toy model of superposition, following, formalizing and extending results from [1], who first gave an empirical demonstration of superposition in a neural network toy model. The toy model is constructed to be one of the simplest possible models where superposition can emerge—a quasi-autoencoder with a bottleneck that forces compression of more features than the number of hidden dimensions.

This chapter addresses three fundamental questions:
(i) Under what conditions does superposition become advantageous?
(ii) What geometric structures emerge when models learn superposed representations?
(iii) How robust is superposition to architectural and optimization choices?

We proceed in three parts. In Section 3.1 we analyze the loss theoretically to understand when superposition emerges. Next, in Section 3.2 we characterize regular geometric solutions—particularly regular polygons—that arise from this optimization. Finally, in Section 3.3 we empirically validate these theoretical predictions and examine robustness of the emergence of superposition.

Note 3.1. (Contribution):
The analysis in Section 3.1 follows results from [1] directly, although we provide additional formalization. Particularly, we prove Proposition 3.8 and Corollary 3.10, which were stated without proof in [1], where the authors do not use the full loss form but an approximation only. The analysis in Section 3.2 as well as Appendix A.4.3 are original contributions from us.

For an overview of notation, see Appendix A.1.

3.1 Toy Model Definition and Loss Analysis

In this section, following [1], we give the toy model setup and analyze its loss function. We formalize derivations from [1], demonstrating that superposition becomes advantageous for the nonlinear model.

3.1.1 Toy Model Setup

To study superposition in isolation, we use the toy model from [1], which satisfies three central properties. First, it uses a bottleneck architecture that forces compression (Definition 3.2). Second, it weighs the loss by feature importance to model realistic scenarios where some features are more important than others (Definition 3.3). Third, it uses sparse inputs, without which superposition would not be advantageous (Definition 3.4).

Definition 3.2. (Toy model of feature representation [1, p. 10]):
The (coupled) toy model is defined as

$$\hat{X} := \alpha(\boldsymbol{W}^T \boldsymbol{W} X + b),$$

where α is an activation function and $\boldsymbol{W} \in \mathbb{R}^{n_h \times m}$ is a matrix mapping from a higher dimensional space $\mathbb{R}^m$ to a lower dimensional hidden one, $\mathbb{R}^{n_h}$, adding a bias vector b and passing through an activation function α. Here, m denotes the number of features and $n_h < m$ the number of hidden dimensions. If we do not specify otherwise, we choose $\alpha := \mathrm{ReLU}$ as activation function.

Notably, due to the coupling of the weight matrices, the read-off vectors are the weight vectors themselves. This coupling simplifies the analysis of the model, as we only have to consider one weight matrix.

In practice, we think of represented features as having different importances: For example, representing a 'clothes' feature likely has a larger importance for distinguishing between humans and apes than general 'nose' or 'eyes' features, although they also likely give some benefit.

To approximate the different importance of different features, we use a weighted loss in the toy model.

Definition 3.3. (Importance-weighted loss [1, p. 10]):
Given *importances* $I_1, ..., I_m$, the toy model loss is defined as

$$L := \sum_i I_i \left(\hat{X}_i - X_i\right)^2.$$

We occasionally use the importances in diagonal matrix form, $\boldsymbol{I}_{ij} := I_i 1_{i=j}$.

As discussed in Chapter 2, sparsity is necessary for superposition. To model sparsity in the toy model, we use a sparse uniform distribution.

Definition 3.4. (Sparse uniform input features [1, p. 9]):
Let $0 \leq S < 1$ be the sparsity factor. Let $U_i \sim \mathcal{U}[0, 1]$ be independent uniform random variables and $B_i \sim \texttt{Ber}(1 - S)$ be independent Bernoulli random variables. Then, the sparse uniform input feature random variable $X \in \mathbb{R}^m$ is defined by

$$X_i := U_i B_i,$$

i.e. each feature is active with probability $1 - S$ and when active, uniformly drawn from $(0, 1]$.

We define $\mathbb{P}_{\text{sparse}}$ as the induced probability distribution of X.

Remark 3.5.
Here, the sparsity (factor) S is a probabilistic parameter (probability that a feature is inactive). Particularly it is not the same as the sparsity of X as a vector from Definition 2.6, which is the number of non-zero elements. To clarify the distinction, we often phrase in terms of feature probability $1 - S$ when discussing the sparsity factor.

Remark 3.6. (Sparse Boolean features):
The article [1] uses simplifications of the integrals in the uniform continuous case by substituting 1 for illustrative purposes, which simplifies many results, although the results differ from the continuous case.

This can be more rigorously understood as the loss of a Boolean feature model, substituting the uniform distribution with a δ_1-distribution, i.e.,

$$X_i = B_i,$$

for $B_i \sim \texttt{Ber}(1-S)$ independently for each feature index i.

We can now illustrate examples of weight configurations learned by the toy model in two dimension, shown in Figure 2.[3]

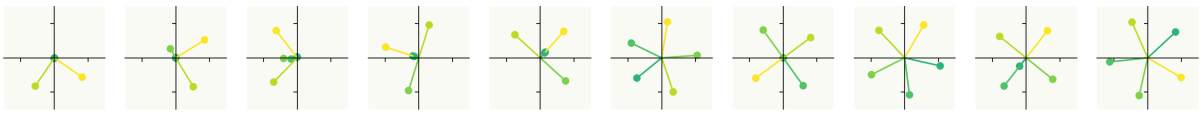


Figure 2: Plot of the (column) weight vectors of the toy model with $m = 5$ features and $n_h = 2$ hidden dimensions with feature probability decreasing from 1 on the left to 0.05 on the right. Most important feature marked in yellow, least important in dark green. Reproduced from [1, p. 2]

Notably, we see that for dense features on the left, the model uses orthogonal weight vectors to represent the features without superposition. Gradually, we see that the less important features then become represented in opposite directions. With even sparser features on the right, we see that the model uses regular polygon configurations to represent the features.

All instances where the vectors are not orthogonal show representational superposition, as more features are represented than there are hidden dimensions.

3.1.2 Loss Function Analysis

From Figure 2, we see that superposition emerges for sufficiently sparse features. To obtain a more precise model of when superposition becomes advantageous, we analyze the expected loss. Our strategy is to decompose the loss by the number of simultaneously active features. This reveals that as feature probability decreases ($S \to 1$), the 1-sparse case dominates, allowing us to derive a tractable analytical form of the benefit-interference tradeoff.

Lemma 3.7. (Decomposition as mixture):
We can decompose the sparse uniform distribution as mixture of k-sparse losses, i.e.,

$$\mathbb{P}_{\text{sparse}} = \sum_k \binom{m}{k} (1-S)^k S^{m-k} P_k,$$

where P_k is the probability distribution on k-sparse vectors, i.e.,

$$P_k(\mathcal{A}) = \mathbb{P}_{\text{sparse}}(\mathcal{A} \mid \|X\|_0 = k).$$

[3] We use the AdamW optimizer with a learning rate of 10^{-3} and default weight decay of $\lambda_{\text{wd}} = 10^{-2}$ and train for $n_{\text{it}} = 10^4$ iterations, where we draw a batch of $n_{\text{batch}} = 1024$ at each iteration. (We approximate the 'infinite data limit' by drawing new samples in each training iteration.) We use exponentially decaying importances, i.e., $I_i = 0.9^{i-1}$, and exponentially decaying feature probabilities, i.e., $(1-S) = \left(\frac{1}{20}\right)^{\frac{i-1}{n_{\text{instances}}-1}}$ for the i-th image from the left.

Proof.
Decomposing the distribution using the product rule, we have

$$\mathbb{P}_{\text{sparse}}(\mathcal{A}) = \sum_k \underbrace{\mathbb{P}(X \in \mathcal{A} \mid \|X\|_0 = k)}_{=P_k(\mathcal{A})} \underbrace{\mathbb{P}(\|X\|_0 = k)}_{\binom{m}{k}(1-S)^k S^{m-k}},$$

where $\mathcal{A}$ is an event.

□

3.1.3 Linear Model Analysis

We first analyze the linear model ($\alpha(x) = x$) to establish a baseline. As we will show, superposition is never advantageous for linear models (Corollary 3.10).

Proposition 3.8. (Loss decomposition into feature benefit and interference for the linear model): The expected loss of the linear coupled model $\boldsymbol{W}^T\boldsymbol{W}X$ for the uniform sparse distribution is

$$\mathbb{E}_{\mathbb{P}_{\text{sparse}}}\left[\left\|\sqrt{\boldsymbol{I}}(\boldsymbol{W}^T\boldsymbol{W}X - X)\right\|_2^2\right] =$$

$$\frac{1}{3}(1-S)\left(\sum_i \underbrace{I_i\left(1 - \left\|W^i\right\|_2^2\right)^2}_{\text{Feature Benefit}} + \underbrace{\sum_i \sum_{j\neq i} I_j (W^i \cdot W^j)^2}_{\text{Interference}}\right) + \mathcal{O}((1-S)^2),$$

where W^i denotes the i-th column of $\boldsymbol{W}$, and $\mathcal{O}$ is w.r.t. $S \to 1$.

The second-order term is non-negative. It is non-zero in general, but vanishes for a weight matrix with only orthogonal or zero columns.

This result shows that the linear model loss without sparsity limit contains two competing terms, the so-called *feature benefit* and *interference*. When representing one more feature, the loss is lowered by the *feature benefit* but rises due to the interference when representing the feature non-orthogonally to existing feature directions.

Proof.
Expanding, we obtain

$$\mathbb{E}_{\mathbb{P}_{\text{sparse}}}\left[\left\|\sqrt{\boldsymbol{I}}(\boldsymbol{W}^T\boldsymbol{W}X - X)\right\|_2^2\right] = \mathbb{E}[X^T(\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W})\boldsymbol{I}(\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W})X].$$

Notably, we have an expectation of the form $\mathbb{E}[X^T\boldsymbol{A}X]$, which decomposes into diagonal and off-diagonal terms

$$\mathbb{E}[X^T\boldsymbol{A}X] = \sum_i \mathbb{E}[X_i^2]A_{ii} + \sum_{i\neq j} \mathbb{E}[X_iX_j]A_{ij}.$$

Because all features X_i are identically distributed, we have $\mathbb{E}[X_1^2] = \mathbb{E}[X_2^2] = \cdots = \frac{1}{3}(1-S)$, and the diagonal terms become

$$\begin{aligned}\mathbb{E}_{\mathbb{P}_{\text{sparse}}}[X_1^2]\operatorname{tr}(\boldsymbol{A}) &= \frac{1}{3}(1-S)\operatorname{tr}\left(\boldsymbol{I}(\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W})^2\right) \\ &= \frac{1}{3}(1-S)\sum_i \left[I_i\left(1 - \left\|W^i\right\|_2^2\right)^2 + \sum_{j\neq i} I_j(W^i \cdot W^j)^2\right],\end{aligned}$$

which is the dominating loss term. For the off-diagonal term, we obtain

$$\mathbb{E}[X_i X_j] \sum_{i \neq j} A_{ij} = \frac{1}{4}(1-S)^2 \ \mathcal{O}(1),$$

which yields a more complex expression in terms of $\boldsymbol{W}$.

If we have a weight matrix with orthogonal and zero columns, i.e., $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{B}$ for a diagonal matrix $\boldsymbol{B}$, $\boldsymbol{A}$ simplifies. We obtain

$$\boldsymbol{A} = (\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W})\boldsymbol{I}(\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W}) = (\mathbf{1} - \boldsymbol{B})\boldsymbol{I}(\mathbf{1} - \boldsymbol{B})$$

which as a product of diagonal matrices is also a diagonal matrix, i.e., all off-diagonal terms vanish and thus the full $\mathcal{O}((1-S)^2)$ term.

□

Note 3.9. (Form of the result):
In the original article [1], the authors state the loss to be proportional to the dominant loss term, quoting a modified version from Saxe [37] without proof.

We expand the exact form of the equation in 2D to show that the form they quote is only an approximation and in general there are more complex mixed 4-th order in W-terms. (They claim based on this result: "In fact, this makes it never worthwhile for the linear model to represent more features than it has dimensions." [1, p.13]. Although their version of the loss is only approximate, we can prove this claim for the full non-approximate version of the loss, which we do in Corollary 3.10.)

As a counterexample why the full loss can in general *not* be written as a sum of purely feature benefit and interference terms, we consider the case of $m = 3$, $n_h = 2$, and homogeneous importances, $\boldsymbol{I} = \mathbf{1}$. In the proof, we look at one off-diagonal term from $\boldsymbol{A}$: $\boldsymbol{A}$ decomposes as

$$\boldsymbol{A} = (\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W})(\mathbf{1} - \boldsymbol{W}^T\boldsymbol{W}) = \mathbf{1} - 2\boldsymbol{W}^T\boldsymbol{W} + (\boldsymbol{W}^T\boldsymbol{W})^2.$$

The full $\mathcal{O}((1-S)^2)$ term is

$$\frac{1}{4}(1-S)^2 \sum_{i \neq j} A_{ij} = \frac{1}{4}(1-S)^2(A_{12} + A_{21} + A_{23} + \cdots)$$

Expanding for example the off-diagonal term A_{12} yields

$$A_{12} = (\ldots) + (\boldsymbol{W}^T\boldsymbol{W})^2_{12} = (\ldots) + \sum_k W^1 \cdot W^k W^2 \cdot W^k = (\ldots) + W^1 \cdot W^3 W^2 \cdot W^3,$$

i.e., the sum contains a term that cannot appear in the form of a feature benefit or interference term because it is a product containing three distinct weight vectors. The term does not cancel out because all other subterms in the $(1-S)^2$ term share the positive sign.

Corollary 3.10. (Superposition is not advantageous for linear models):
The global minimum of the loss for the linear coupled model is attained for $\boldsymbol{W}$ consisting of n_h fully orthogonal columns for the most important directions and zero columns.

Proof.
We first consider the dominant term of the loss, which has a nice representation in terms of the

Frobenius norm and then extend the result for the full loss. The dominant term from Proposition 3.8 up to constant factors can be written as the Frobenius norm in terms of the square root of the importance matrix, i.e.,

$$\text{Dominant Loss Term} \propto \left\| \sqrt{\boldsymbol{I}}(\mathbf{1} - \boldsymbol{W}^T \boldsymbol{W}) \right\|_F^2$$

Taking the singular value decomposition of $\boldsymbol{W}$, $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, we can rewrite the dominant term to obtain a representation that explicitly depends on the singular values of $\boldsymbol{W}$. Thus, we have

$$\begin{aligned}
&\left\| \sqrt{\boldsymbol{I}}(\mathbf{1} - \boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T) \right\|_F^2 && \downarrow \text{Factor out } \boldsymbol{V} \\
&= \left\| \sqrt{\boldsymbol{I}}\boldsymbol{V}(\mathbf{1} - \boldsymbol{\Sigma}^T\boldsymbol{\Sigma})\boldsymbol{V}^T \right\|_F^2 && \downarrow \text{Invariance of } \|\cdot\|_F \text{under orthonormal } \boldsymbol{V}^T \\
&= \left\| \sqrt{\boldsymbol{I}}\boldsymbol{V}(\mathbf{1} - \boldsymbol{\Sigma}^T\boldsymbol{\Sigma}) \right\|_F^2 && \downarrow \text{Trace representation} \\
&= \operatorname{tr}\left(\left(\sqrt{\boldsymbol{I}}\boldsymbol{V}(\mathbf{1} - \boldsymbol{\Sigma}^T\boldsymbol{\Sigma}) \right)^T \left(\sqrt{\boldsymbol{I}}\boldsymbol{V}(\mathbf{1} - \boldsymbol{\Sigma}^T\boldsymbol{\Sigma}) \right) \right) && \downarrow \text{Cyclic permutation property} \\
&= \operatorname{tr}\left(\boldsymbol{V}^T\boldsymbol{I}\boldsymbol{V}(\mathbf{1} - \boldsymbol{\Sigma}^T\boldsymbol{\Sigma})^2 \right) && \downarrow \text{Trace expansion} \\
&= \sum_i \underbrace{(\boldsymbol{V}^T\boldsymbol{I}\boldsymbol{V})_{ii}}_{\geq 0} (1 - \sigma_i^2)^2,
\end{aligned}$$

where $(\boldsymbol{V}^T\boldsymbol{I}\boldsymbol{V})_{ii} = e_i^T(\boldsymbol{V}^T\boldsymbol{I}\boldsymbol{V})e_i \geq 0$ because congruence transformations preserve positive semi-definiteness.

In this representation of the loss, we see that the minimum of the dominant-order term is attained for the n_h most highly weighted singular values (we cannot have more than n_h nonzero singular values, as the rank of $\boldsymbol{W}$ is at most n_h). From Proposition 3.8, we also have that the second-order term vanishes for orthogonal columns, i.e., for example, for the matrix $\boldsymbol{W} \coloneqq \mathbf{1}_{n_h} \in \mathbb{R}^{n_h \times m}$, which is a minimum of the dominant-order term and thus also a global minimum of the loss.

□

3.1.4 Non-Linear Model Analysis

Notably, we can prove an analogous result for the sparse limit of the nonlinear model loss, which illustrates that it actually becomes advantageous for the nonlinear model to represent features in superposition, in contrast to the linear case established in Corollary 3.10.

Proposition 3.11. (Loss decomposition of the non-linear ReLU model [1, p.13]):
The expected loss decomposes into terms for different numbers of active features, i.e.,

$$l = \mathbb{E}_{\mathbb{P}_{\text{sparse}}}\left[\sum_i I_i\left(\hat{X}_i - X_i\right)^2\right] = \sum_{k=0}^{m}(1-S)^k S^{m-k} l_k, \tag{3.1}$$

where l_k is the sum of expected losses for all patterns with exactly k active features.

Here, the loss for $k=0$ is $\sum_i I_i \mathrm{ReLU}(b_i)^2$, a penalty for positive biases, and the loss term l_1 becomes

$$l_1 = \sum_i \underbrace{I_i\ \mathbb{E}\left[\left(Y - \mathrm{ReLU}\left(\left\|W^i\right\|_2^2 Y + b_i\right)\right)^2\right]}_{\text{Feature Benefit}} + \sum_i \sum_{j\neq i} \underbrace{I_j\ \mathbb{E}\left[\mathrm{ReLU}\left(W^j \cdot W^i Y + b_j\right)^2\right]}_{\text{Interference}} \tag{3.2}$$

where Y is the distribution of an active feature, i.e. $Y \sim \mathbb{P}(X_1 \mid X_1 > 0)$, and W^i denotes the i-th column of $\boldsymbol{W}$.

For Boolean features, this term reduces to

$$l_1^{\text{Bool}} = \sum_i I_i\left(1 - \mathrm{ReLU}\left(\left\|W^i\right\|_2^2 + b_i\right)\right)^2 + \sum_i \sum_{j\neq i} I_j \mathrm{ReLU}\left(W^j \cdot W^i + b_j\right)^2. \tag{3.3}$$

Note 3.12.
Comparing the interference to the interference term in the linear case, we notice that negative interference is eliminated by the ReLU-function in the 1-sparse case.

Expanding binomially in terms of $(1-S)$ and considering the limit $S \to 1$, the total expected loss becomes

$$l(S) = S^m l_0 + S^{m-1}(1-S) l_1 + \mathcal{O}((1-S)^2).$$

Thus, for high sparsities, the 1-sparse loss term dominates the loss after the bias penalty term l_0.

Because of the similarity to packing points in space, minimizing a potential, (3.3) is called a modified Thompson problem in [1].

Proof of Proposition 3.11.
According to Lemma 3.7, the loss decomposes into k-sparse terms

$$\begin{aligned}\mathbb{E}[l] &= \mathbb{E}\left[\sum_i I_i\left(\hat{X}_i - X_i\right)^2\right] \\ &= \sum_k (1-S)^k S^{m-k} \underbrace{\binom{m}{k} \mathbb{E}_{P_k}\left[\sum_i I_i\left(\hat{X}_i - X_i\right)^2\right]}_{l_k},\end{aligned}$$

which is (3.1). Calculating the 0-sparse loss, we have

$$l_0 = \mathbb{E}_{P_0}\left[\sum_i I_i \left(\underbrace{\hat{X}_i}_{\mathrm{ReLU}(b_i)} - \underbrace{X_i}_{0}\right)^2\right]$$
$$= \sum_i I_i \mathrm{ReLU}(b_i)^2.$$

For the 1-sparse loss, we sum the conditional expectations over all features,

$$l_1 = m\mathbb{E}_{X\sim P_1}\left[\left\|\sqrt{\boldsymbol{I}}\left(\hat{X}-X\right)\right\|_2^2\right]$$
$$= \sum_i \mathbb{E}_{X_i\sim\mathcal{U}}\left[\sum_j I_j\left(\hat{X}_j - X_j\right)^2 \mid X_i \neq 0\right]$$
$$= \sum_i \left(I_i \mathbb{E}_{X_i\sim\mathcal{U}}\left[\left(\mathrm{ReLU}(W^i\cdot W^i X_i + b_i) - X_i\right)^2\right] + \sum_{j\neq i} I_j \mathbb{E}_{X_i\sim\mathcal{U}}\left[\mathrm{ReLU}\left(W^j\cdot W^i X_i + b_j\right)^2\right]\right).$$

Because each X_i is identically uniformly distributed, we substitute each by a common variable $Y \sim \mathcal{U}[0,1]$ to obtain (3.2).

Changing the distribution of an active individual feature from a uniform distribution to a δ_1-distribution (which covers the case of Boolean features), i.e. substituting $X_j = 1$ in the expectation, we obtain (3.3).

□

3.2 Regular Geometric Solutions

When training the model, we often observe regular structures. In two dimensions, we often observe regular polygons, as shown in Figure 2, while in higher dimensions, we can observe so-called tegum products of simple figures (Definition 3.16). Notably, symmetric structures such as polygons often also occur in the case of inhomogeneous importances, such as in Figure 2. We characterize these solutions analytically in Section 3.2.1 and analyze them numerically in Section 3.2.2.

3.2.1 Zero-Bias Approximation

We begin by analyzing the simpler case of zero bias with homogeneous importances. While numerical solutions such as the pentagon solutions shown in Figure 2 often have non-zero bias, this approximation yields analytical tractability and insights. We extend to the non-zero bias case numerically in Section 3.2.2 and investigate the stability of these solutions empirically in Section 3.3.

Under these assumptions, we can prove that regular polygons are stationary points of the l_1-loss defined in (3.2). We prove a general formula characterizing stationary solutions with zero bias and homogeneous importances.

Proposition 3.13. (Stationary Point Condition):
Let a configuration $(\boldsymbol{W}, b = 0)$ with zero bias and with nonzero weight columns $W^i \neq 0$ be given, where W^i denotes the i-th column of $\boldsymbol{W}$. This configuration is a stationary point of the l_1 expected loss defined in (3.2) if and only if

$$-\left(1 - \left\|W^i\right\|_2^2\right) W^i + \sum_{j \in C(i)} W^j \cdot W^i W^j = 0$$

holds for all i. Here $C(i) := \{j \in 1{:}m \mid W^i \cdot W^j > 0, j \neq i\}$ is the set of all indices with weight vectors in the positive cone of W^i, i.e., where i, j have a relative angle of less than $\frac{\pi}{2}$.

Proof of Proposition 3.13.
Substituting $b = 0$, and $I_i = 1$ equation (3.2) becomes

$$\begin{aligned} l_1 &= \sum_i \mathbb{E}_{Y \sim \mathcal{U}} \left[\left(Y - \mathrm{ReLU}\left(\left\|W^i\right\|_2^2 Y \right) \right)^2 \right] + \sum_i \sum_{j \neq i} \mathbb{E}_{Y \sim \mathcal{U}} \left[\mathrm{ReLU}(W^j \cdot W^i Y)^2 \right] \\ &= \sum_i \mathbb{E}_{Y \sim \mathcal{U}} \left[\left(Y - \left(\left\|W^i\right\|_2^2 Y \right) \right)^2 \right] + \sum_i \sum_{j \in C(i)} \mathbb{E}_{Y \sim \mathcal{U}} \left[(W^j \cdot W^i Y)^2 \right], \end{aligned}$$

where we used that the ReLU-argument in the feature benefit term is nonnegative because $Y, \left\|W^i\right\|_2 \geq 0$ and for the second term, we have $W^j \cdot W^i Y \geq 0$ precisely for $j \in C(i)$. We now pull the expectation out of the sum, yielding

$$l_1 = \sum_i \mathbb{E}_{Y \sim \mathcal{U}}[Y^2] \left(1 - \left\|W^i\right\|_2^2\right)^2 + \sum_i \sum_{j \in C(i)} \mathbb{E}_{Y \sim \mathcal{U}}[Y^2] (W^j \cdot W^i)^2.$$

Taking the gradient w.r.t. a column W^l, we obtain

$$\nabla_{W^l} l_1 = -4 \mathbb{E}_{Y \sim \mathcal{U}}[Y^2] \left(1 - \left\|W^l\right\|_2^2\right) W^l + 2 \sum_{j \in C(l)} \mathbb{E}_{Y \sim \mathcal{U}}[Y^2] 2 W^j \cdot W^l W^j,$$

where we have an additional factor of 2 because for $j \in C(l)$ there are two terms including W^l in the double sum, one with $i = l$ and one with $j = l$.

Dividing by $4\mathbb{E}_{Y \sim \mathcal{U}}[Y^2]$, we obtain the stationary point condition in the proposition.

□

Notably, the stationary point condition Proposition 3.13 also holds for different nonnegative feature distributions, e.g. the Boolean feature distribution Remark 3.6, as we can see by substituting all uniform expectations with the expectation over the constant $Y \equiv 1$ distribution for an active feature.

To prove that regular polygons are stationary points, we first establish two geometric properties of polygonal vector sets. Equation (3.4) describes a symmetry: opposite vertices sum to a scaled version of any vertex, simplifying interference calculations. The second property (3.5) characterizes how many neighbors contribute to interference.

Lemma 3.14. (Properties of regular polygons):
Let a regular m-gon with radius r be defined as $W^i := r(\cos(\Delta\varphi\ i), \sin(\Delta\varphi\ i))^T$ for $i \in \mathbb{Z}$, where $\Delta\varphi := \frac{2\pi}{m}$ is the angle between two vertices of the polygon, and W^i denotes the i-th column of the corresponding weight matrix $\boldsymbol{W}$. Then, the following properties hold:

(i) The sum of two "opposite" vertices $W^{i-\Delta\mathrm{i}} + W^{i+\Delta\mathrm{i}}$ yields a multiple of the vector W^i, specifically

$$W^{i-\Delta\mathrm{i}} + W^{i+\Delta\mathrm{i}} = 2\cos(\Delta\varphi\Delta\mathrm{i})W^i. \tag{3.4}$$

(ii) The number of vertices in the positive cone for any vertex is $2k$, where

$$k = \left\lfloor \frac{m-1}{4} \right\rfloor,$$

i.e.,

$$\left|\{j \mid W^i \cdot W^j > 0, j \neq i\}\right| = 2k. \tag{3.5}$$

Proof.
The first property follows from the addition laws of trigonometry: For the first coordinate, we have

$$\begin{aligned}
\left(W^{i-\Delta\mathrm{i}} + W^{i+\Delta\mathrm{i}}\right) \cdot e_1 &= r(\cos(\Delta\varphi\ (i-\Delta\mathrm{i})) + \cos(\Delta\varphi\ (i+\Delta\mathrm{i}))) \\
&= r(\cos(\Delta\varphi\ i)\cos(\Delta\varphi\Delta\mathrm{i}) - \sin(\Delta\varphi\ i)\sin(\Delta\varphi\Delta\mathrm{i}) \\
&\quad + \cos(\Delta\varphi\ i)\cos(\Delta\varphi\Delta\mathrm{i}) + \sin(\Delta\varphi\ i)\sin(\Delta\varphi\Delta\mathrm{i})) \\
&= 2r\cos(\Delta\varphi\Delta\mathrm{i})\cos(\Delta\varphi\ i),
\end{aligned}$$

and for the y coordinate, we have

$$\begin{aligned}
\left(W^{i-\Delta\mathrm{i}} + W^{i+\Delta\mathrm{i}}\right) \cdot e_2 &= r(\sin(\Delta\varphi\ (i-\Delta\mathrm{i})) + \sin(\Delta\varphi\ (i+\Delta\mathrm{i}))) \\
&= r(\sin(\Delta\varphi\ i)\cos(\Delta\varphi\Delta\mathrm{i}) - \cos(\Delta\varphi\ i)\sin(\Delta\varphi\Delta\mathrm{i}) \\
&\quad + \sin(\Delta\varphi\ i)\cos(\Delta\varphi\Delta\mathrm{i}) + \cos(\Delta\varphi\ i)\sin(\Delta\varphi\Delta\mathrm{i})) \\
&= 2r\cos(\Delta\varphi\Delta\mathrm{i})\sin(\Delta\varphi\ i),
\end{aligned}$$

which gives the first property.

For the second claim, we note that the angle between two vertices is $\Delta\varphi = \frac{2\pi}{m}$ and the positive cone is defined as the set of all indices for vertices with an angle of less than $\frac{\pi}{2}$. So, the number k satisfies $k\frac{2\pi}{m} < \frac{\pi}{2}$ and $(k+1)\frac{2\pi}{m} \geq \frac{\pi}{2}$. Multiplying by $2\frac{m}{\pi}$ gives $k < \frac{m}{4}$ and $(k+1) \geq \frac{m}{4}$, which is satisfied by $k = \left\lfloor \frac{m-1}{4} \right\rfloor$.

□

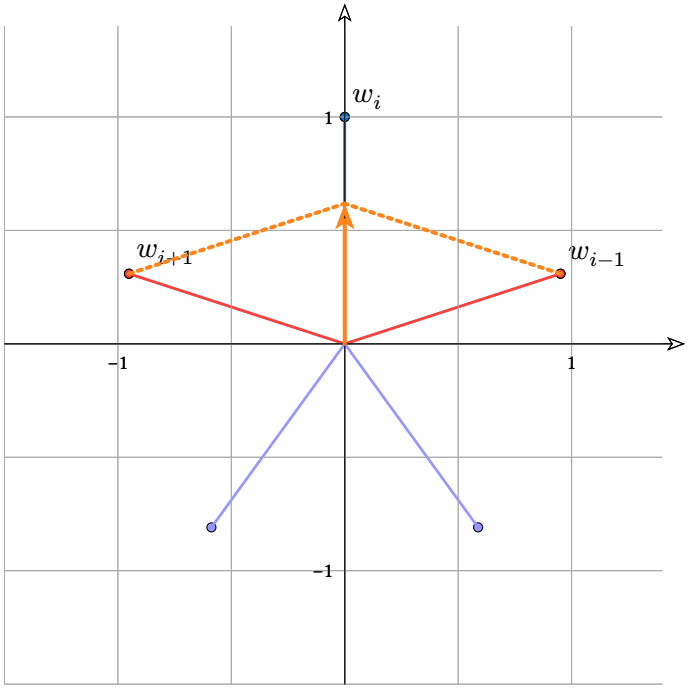


Figure 3: Illustration of property (3.4). Left: A regular pentagon with highlighted vertices, with W^{i+1} and W^{i-1} summing to $2\cos(\Delta\varphi\Delta\mathrm{i})W^i$, marked in orange.

Corollary 3.15. (Regular polygon solutions are stationary points):
Define the regular polygon solution as $W^i = r(\cos(\Delta\varphi\ i), \sin(\Delta\varphi\ i))^T$ for $i \in 1{:}m$, where $\Delta\varphi := \frac{2\pi}{m}$ is the angle between two vertices of the polygon, and W^i denotes the i-th column of $\boldsymbol{W}$. Let $C_k := \sum_{1\le j\le k} \cos(\Delta\varphi\ j)^2$, where $k = \left\lfloor \frac{m-1}{4} \right\rfloor$. The regular polygon solution with $b = 0$ is a stationary point of the zero-bias l_1-loss from (3.2), where r is the norm of each vector given by

$$r^2 = \frac{1}{1 + 2C_k}.$$

Proof.
For this proof, as in Lemma 3.14, we use integer indices for the vertices of the polygon, i.e. we set $W^i := r(\cos(\Delta\varphi\ i), \sin(\Delta\varphi\ i))^T$ for all $i \in \mathbb{Z}$, i.e., the indices are understood cyclically.

Now, evaluating the stationary point condition, we have

$$\begin{aligned}
0 &= -\left(1 - \left\|W^i\right\|_2^2\right) W^i + \sum_{j \in C(i)} W^j \cdot W^i W^j \\
&= -(1 - r^2) W^i + \sum_{1 \le \Delta\mathrm{i}\ \le k} \underbrace{W^{i+\Delta\mathrm{i}} \cdot W^i}_{= r^2 \cos(\Delta\varphi\Delta\mathrm{i})} W^{i+\Delta\mathrm{i}} + W^{i-\Delta\mathrm{i}} \cdot W^i W^{i-\Delta\mathrm{i}} \\
&= -(1 - r^2) W^i + \sum_{1 \le \Delta\mathrm{i}\ \le k} 2r^2 \cos\left(\Delta\varphi\Delta\mathrm{i}\right)^2 W^i,
\end{aligned}$$

where we see that this equation is satisfied if

$$r^2 - 1 + \sum_{1 \le \Delta\mathrm{i}\ \le k} 2r^2 \cos\left(\Delta\varphi\Delta\mathrm{i}\right)^2 = 0,$$

which yields the equation in the corollary.

□

We tabulate the values of $C_k(m)$ for different m. We see that, starting at $m = 5$, the radiuses of the zero-bias regular polygon solutions decrease as compounding positive interference increases.

Notably, this is qualitatively different from the proper, non-zero bias case, where the radii first increase before decreasing again, compensated by negative bias terms, as we will see in Section 3.2.2. Thus, geometric symmetry of solutions transfers to the non-zero bias case, the behavior of radii and losses differ.

Table 1: Values for the cosine sum coefficient $C_k = \sum_{j=1}^{k} \cos^2\left(\frac{2\pi j}{m}\right)$, the squared radius r^2 of the regular polygon solution, and the corresponding l_1-loss for Boolean features Corollary 3.15 for different numbers of vertices m.

$m = 4$	$m = 5$	$m = 6$	$m = 7$	$m = 8$	$m = 9$	$m = 10$
$C_k = 0$	$C_k = 0.095$	$C_k = 0.25$	$C_k = 0.389$	$C_k = 0.5$	$C_k = 0.617$	$C_k = 0.75$
$r^2 = 1$	$r^2 = 0.84$	$r^2 = 0.667$	$r^2 = 0.563$	$r^2 = 0.5$	$r^2 = 0.448$	$r^2 = 0.4$
${l_1}^{\mathrm{B}} = 0$	${l_1}^{\mathrm{B}} = 0.802$	${l_1}^{\mathrm{B}} = 2$	${l_1}^{\mathrm{B}} = 3.062$	${l_1}^{\mathrm{B}} = 4$	${l_1}^{\mathrm{B}} = 4.971$	${l_1}^{\mathrm{B}} = 6$

In the article, the authors find that in higher dimensions, solutions are often not regular polytopes, but so-called tegum products or direct sums of regular polytopes or polygons.

Definition 3.16. (Direct sum, tegum product [38]):
The *direct sum* or tegum product $P_1 \oplus P_2$ of two polytopes $P_1 \subset \mathbb{R}^{n_1}, P_2 \subset \mathbb{R}^{n_2}$ with vertices $V(P_1), V(P_2)$ is defined as the convex hull of the points of the polygons placed in orthogonal subspaces, i.e.,

$$P_1 \oplus P_2 = \text{cvxhull}(\{(v_1, 0) \mid v_1 \in V(P_1)\} \cup \{(0, v_2) \mid v_2 \in V(P_2)\}).$$

For example, we can construct pyramids out of regular polygons, such as the pentagonal bipyramid, given as a direct sum of a pentagon and a digon, i.e., $P_5 \oplus [-1, 1]$, where $P_5 = \text{cvxhull}\left(\left\{\left(\cos\left(2\pi\frac{i}{5}\right), \sin\left(2\pi\frac{i}{5}\right)\right)^T \mid i \in 0{:}4\right\}\right)$ is the pentagon and $[-1, 1]$ is the digon in $\mathbb{R}^1$.

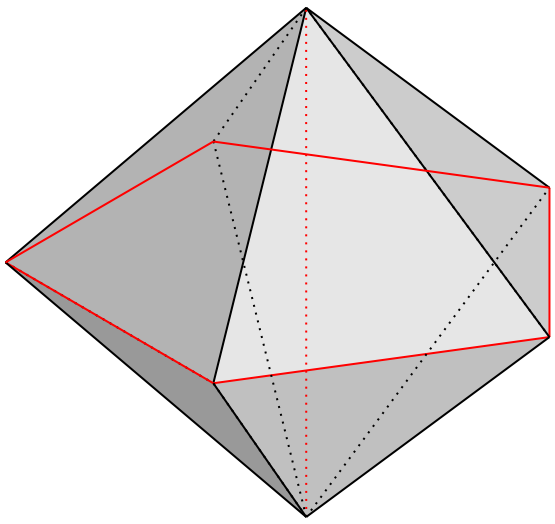


Figure 4: A pentagonal bipyramid, a direct sum of a pentagon and a digon [39].

Corollary 3.17. (Stationarity of tegum product solutions):
If $\boldsymbol{W} \in \mathbb{R}^{d_1 \times n_1}$, $\boldsymbol{V} \in \mathbb{R}^{d_2 \times n_2}$ with zero bias are both stationary points of the l_1-loss defined in (3.2) (for the respective configurations), then the "tegum product" matrix defined by

$$\boldsymbol{U} := \left(\begin{pmatrix} W^1 \\ 0 \end{pmatrix}, ..., \begin{pmatrix} W^{n_1} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ V^1 \end{pmatrix}, ..., \begin{pmatrix} 0 \\ V^{n_2} \end{pmatrix} \right) \in \mathbb{R}^{(d_1+d_2)\times(n_1+n_2)}$$

is also a stationary point of the l_1-loss defined in (3.2), where $\begin{pmatrix} x \\ y \end{pmatrix}$ is the vector concatenation of the two vectors x, y, and W^i, V^j denote the i-th and j-th columns of $\boldsymbol{W}$ and $\boldsymbol{V}$, respectively.

Proof.
In Proposition 3.13, the stationary point condition holds for every U^i: Because W and V are embedded in orthogonal subspaces, the set of neighbors of $\begin{pmatrix} W^i \\ 0 \end{pmatrix}$ is the previous set of neighbors of W^i (the angles between vectors in orthogonal subspace is $\frac{\pi}{2}$), and the natural embedding $v \mapsto (v, 0)$ preserves the inner products.

□

3.2.2 General Bias Case

The zero-bias analysis provided analytical insights, but the toy model bias is non-zero in many cases. Direct optimization over all weights and biases is intractable analytically. However, we can exploit the symmetry of polygon solutions: due to rotational invariance, the loss depends only on the radius r and uniform bias β. This reduces the optimization to a 2D *polygon manifold* $\mathcal{P}$, which we can analyze both analytically and numerically.[4]

For a fixed number of vertices m, we parameterize the polygon manifold by

$$P(r, \beta) := \left(\begin{pmatrix} r\cos(\Delta\varphi\ i) \\ r\sin(\Delta\varphi\ i) \end{pmatrix}_{i \in 1:m}, \beta(1, ..., 1) \right) \in \mathbb{R}^{2\times m} \times \mathbb{R}^m, P : \mathbb{R}^+ \times \mathbb{R} \to \mathcal{P},$$

where $\Delta\varphi := \frac{2\pi}{m}$ is the angle between adjacent vertices of the polygon. The main insight, formalized in the following lemma, is that the gradient always lies in the tangent space of this manifold.

Lemma 3.18. (Gradient of the l_1-loss is tangential to the polygon manifold):
The gradient of the l_1-loss (3.2) for a polygon solution $(\boldsymbol{W}, b)$ with non-zero bias is tangential to the polygon manifold $\mathcal{P}$, i.e.,

$$\nabla l_1(p) \in T_p\mathcal{P},$$

for $p = P(r, \beta)$ with $r > 0, \beta \in \mathbb{R}$.

Proof of Lemma 3.18.
The loss is given by (3.2), which yields for homogeneous importances

$$\begin{aligned} l_1(p = (\boldsymbol{W}, b)) &= \sum_i \mathbb{E}\left[\left(\mathrm{ReLU}\left(\left\|W^i\right\|_2^2 Y + b_i \right) - Y \right)^2 \right] + \sum_i \sum_{j \neq i} \mathbb{E}\left[\mathrm{ReLU}\left(W^j \cdot W^i Y + b_j \right)^2 \right] \\ &= \sum_i \mathbb{E}\left[\left(\mathrm{ReLU}(s_i(Y)) - Y \right)^2 \right] + \sum_i \sum_{j \neq i} \mathbb{E}\left[\mathrm{ReLU}\left(s_{i,j}(Y) \right)^2 \right] \\ &=: B(\boldsymbol{W}, b) + I(\boldsymbol{W}, b), \end{aligned}$$

[4] The code for the numerical analysis can be found in `experimentsjl/polygon_solution_full_nb.jl`.

where we abbreviated the inner terms $s_i(Y) := \left\|W^i\right\|_2^2 Y + b_i$ and $s_{i,j}(Y) := W^j \cdot W^i Y + b_j$ and split the loss into benefit and interference terms. We now collect the gradients of the benefit and interference terms.[5]

$$\begin{aligned}\nabla_{W_i} B &= \mathbb{E}\Big[\mathrm{ReLU}'(s_i(Y))\nabla_{W_i} s_i(Y) 2(\mathrm{ReLU}(s_i(Y)) - Y)\Big] \\ &= 4\ \mathbb{E}\Big[1_{s_i(Y)>0}(s_i(Y) - Y)\Big] W_i,\end{aligned}$$

where we used that $\nabla_{W_i} s_i(Y) = 2W_i$ and $\mathrm{ReLU}'(s_i(Y)) = 1_{s_i(Y)>0}$ is the indicator variable. Here, we omitted the ReLU, because the indicator variable already sets the term to 0 if $s_i(Y) \leq 0$. Similarly, we obtain for the gradient with respect to the bias

$$\begin{aligned}\nabla_{b_i} B &= \mathbb{E}\Big[\mathrm{ReLU}'(s_i(Y))\nabla_{b_i} s_i(Y) 2(\mathrm{ReLU}(s_i(Y)) - Y)\Big] \\ &= 2\ \mathbb{E}\Big[1_{s_i(Y)>0}(s_i(Y) - Y)\Big].\end{aligned}$$

For the interference term, we have

$$\begin{aligned}&\nabla_{W_i} I \\ &= \sum_{j\neq i} \mathbb{E}\Big[2\,\mathrm{ReLU}'\big(s_{i,j}(Y)\big)\nabla_{W_i} s_{i,j}(Y)\,\mathrm{ReLU}\big(s_{i,j}(Y)\big)\Big] \\ &\quad + \mathbb{E}\Big[2\,\mathrm{ReLU}'\big(s_{j,i}(Y)\big)\nabla_{W_i} s_{j,i}(Y)\,\mathrm{ReLU}\big(s_{j,i}(Y)\big)\Big] \\ &= \sum_{j\neq i} 2\ \mathbb{E}\Big[1_{s_{i,j}(Y)>0} s_{i,j}(Y) Y\Big] W_j + 2\ \mathbb{E}\Big[1_{s_{j,i}(Y)>0} s_{j,i}(Y) Y\Big] W_j \\ &= 2\sum_{j\neq i}\Big(\mathbb{E}\Big[1_{s_{i,j}(Y)>0} s_{i,j}(Y) Y + 1_{s_{j,i}(Y)>0} s_{j,i}(Y) Y\Big]\Big) W_j.\end{aligned}$$

For the bias, we have

$$\begin{aligned}\nabla_{b_i} I &= \sum_{j\neq i} \mathbb{E}\Big[2\,\mathrm{ReLU}'\big(s_{i,j}(Y)\big)\nabla_{b_i} s_{i,j}(Y)\,\mathrm{ReLU}\big(s_{i,j}(Y)\big)\Big] \\ &= 2\sum_{j\neq i} \mathbb{E}\Big[1_{s_{i,j}(Y)>0} s_{i,j}(Y)\Big].\end{aligned}$$

Now, we insert the polygon solution $p = (\boldsymbol{W}, b) = P(r, \beta)$. The helper terms become $s^p(Y) := s_i(Y) = r^2 Y + \beta$ and $s^p_{\Delta \mathrm{i}}(Y) := s_{i,j}(Y) = r^2 \cos(\Delta \mathrm{i} \cdot \Delta\varphi) Y + \beta = s_{j,i}(Y)$, where $\Delta \mathrm{i} := i - j$ and $\Delta\varphi := \frac{2\pi}{m}$, and the $s_{i,j}(Y)$ term is symmetric in the polygon case. We denote the weight column vectors of the polygon solutions as $W^i_p := r(\cos(\Delta\varphi\ i), \sin(\Delta\varphi\ i))^T$. This yields the gradients. Due to the symmetry of the polygon solution, where all vectors have the same norm r and are regularly spaced, the interference gradients simplify. By symmetry, each $\nabla_{W_i} I$ is proportional to W^i_p:

$$\begin{aligned}\nabla_{W_i} B &= 4\ \mathbb{E}\Big[1_{s^p(Y)>0}(s^p(Y) - Y)\Big] W^i_p, \\ \nabla_{b_i} B &= 2\ \mathbb{E}\Big[1_{s^p(Y)>0}(s^p(Y) - Y)\Big], \\ \nabla_{W_i} I &= 4 W^i_p \sum_{\Delta \mathrm{i}\ \in 1:m-1} \mathbb{E}\Big[1_{s^p_{\Delta \mathrm{i}}(Y)>0} s^p_{\Delta \mathrm{i}}(Y) Y\Big] \cos(\Delta \mathrm{i} \cdot \Delta\varphi), \\ \nabla_{b_i} I &= 2 \sum_{\Delta \mathrm{i}\ \in 1:m-1} \mathbb{E}\Big[1_{s^p_{\Delta \mathrm{i}}(Y)>0} s^p_{\Delta \mathrm{i}}(Y)\Big].\end{aligned}$$

[5] Differentiation under the integral sign is justified by the Leibniz integral rule, as the integrand is differentiable almost everywhere (the set of non-differentiable points of the ReLU function has measure zero with respect to the distribution of Y) and the partial derivatives are bounded by an integrable function of Y.

The tangent space $T_p\mathcal{P}$ at a polygon solution is spanned by:

(i) The direction of changing the radius: $\frac{\partial}{\partial r}$, which corresponds to scaling all weight vectors uniformly:

$$\frac{\partial}{\partial r}P(r,\beta) = \left(\left(\widehat{W_p^i}\right)_i, 0\right).$$

(ii) The direction of changing the bias: $\frac{\partial}{\partial \beta}$, which corresponds to changing all biases uniformly:

$$\frac{\partial}{\partial \beta}P(r,\beta) = (\mathbf{0}, (1,...,1)^T).$$

We see that the gradients of the benefit and interference terms are linear combinations of the two tangent space directions, i.e.,

- The gradient of each weight vector $\nabla_{W_i}(B+I)$ is a scalar multiple of W_p^i (independent of the specific index i), and
- The gradient of each bias $\nabla_{b_i}(B+I)$ is the same scalar for all i, i.e., a multiple of $(1,...,1)^T$.

Thus, we have $\nabla l_1(p) \in T_p\mathcal{P}$.

□

Corollary 3.19. (Solutions on the polygon manifold are stationary points of the l_1-loss): Any stationary point on the polygon manifold $\mathcal{P}$ is a stationary point of the l_1-loss (3.2).

We now compute solutions using BFGS to find stationary points of the l_1-loss on the polygon manifold from Lemma 3.18, where we analytically integrate the expectations in the loss defined in (3.2).

Notably, when one uses numerical integration, multiple different solutions arise for the same number of features, likely due to numerical inaccuracies and flat loss landscapes. Thus, we analytically integrate the expectation integrals to obtain a solution with higher numerical accuracy. See Appendix A.5.1 for derivation and statement of the expectation-free form of the l_1-loss.

Notably, any solution with sufficiently small bias is a trivial local minimum of the loss, so we only consider non-trivial solutions with $r^2 + \beta > 0$. Considering multiple starting values for r, β, we plot the resulting solutions in terms of $\rho := r^2, \beta$.

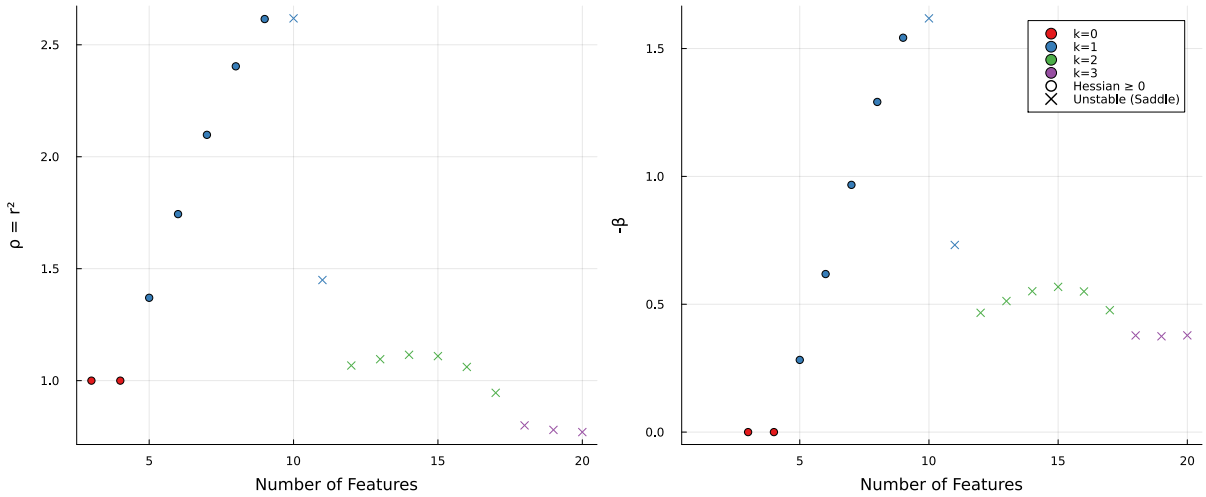


Figure 5: Radiuses and biases found using BFGS plotted against the number of features. Number of interfering neighbors $2k$ denoted by colors. Positive semidefinite (PSD, $H \geq 0$) Hessian solutions marked with circles. Solutions with non-PSD Hessian marked with crosses.

Interestingly, we see a rise and then a decay in the solutions, after which the solutions stay approximately level. Also, from $m = 10$ onwards, we see that the solutions become unstable, i.e., have negative eigenvalues in the Hessian.

The parameters ρ and β appear strongly correlated with nearly identical curves. To understand this relationship, we examine the ratio $\gamma := -\frac{\beta}{\rho}$. This quantity has natural geometric interpretation: it determines the activation threshold in the loss integrals, which in turn controls how many neighboring features contribute interference. For example, if $\gamma > 1$, the solution is trivial (all ReLUs inactive); if $1 > \gamma > \cos\left(\frac{2\pi}{m} \cdot 1\right)$, we have 0 interfering neighbors; if $\cos\left(\frac{2\pi}{m} \cdot 1\right) > \gamma > \cos\left(\frac{2\pi}{m} \cdot 2\right)$, we have 2 interfering neighbors ($k = 1$). Thus γ is a fundamental parameter governing solution structure.

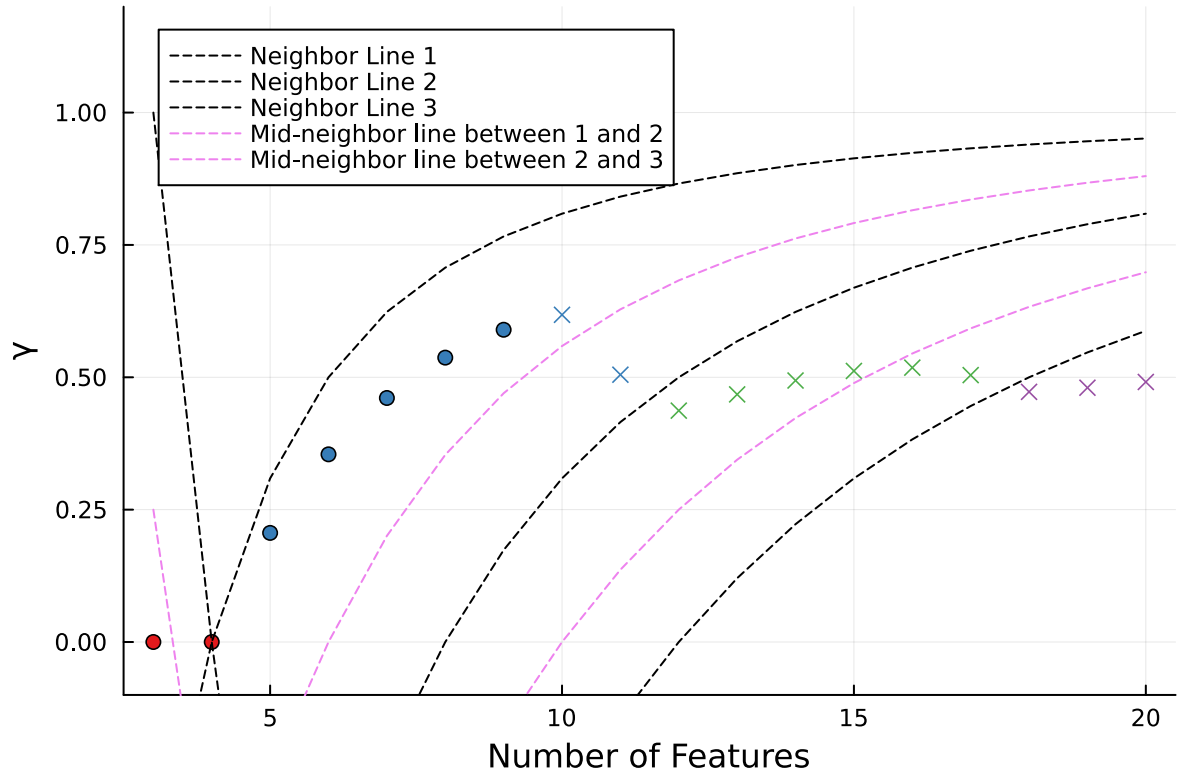


Figure 6: The bias-radius ratio $\gamma := -\frac{\beta}{\rho} = -\frac{\beta}{r^2}$ plotted against m. Neighbor boundary lines marked as dashed lines in black. For example $k = 1$ means $2k = 2$ interfering neighbor vectors for each W^i

We see that γ values of the stable $k = 1$ solutions seem to lie on a continuous curve; similarly for the green solutions. The change of stability seems to happen approximately in the middle between the neighbor boundaries.

In order to investigate the behavior of the solutions with a higher level of detail, we now use a derived continuous representation. We fully evaluate the l_1-loss by analytic integration of the expectations, which yields a rational function in terms of ρ, γ, which depends continuously on $\left(\cos\left(\frac{2\pi}{m} \cdot k\right)\right)_k$ as derived from Lemma 3.14. Fixing the number of interfering neighbors, $2k$, we obtain a functional representation for each k, such that we can extend the solutions even to previously invalid regions. The full computation can be found in Appendix A.5.2. We use BFGS with the same starting values as before to find local (ρ, β)-minima of the l_1-loss.

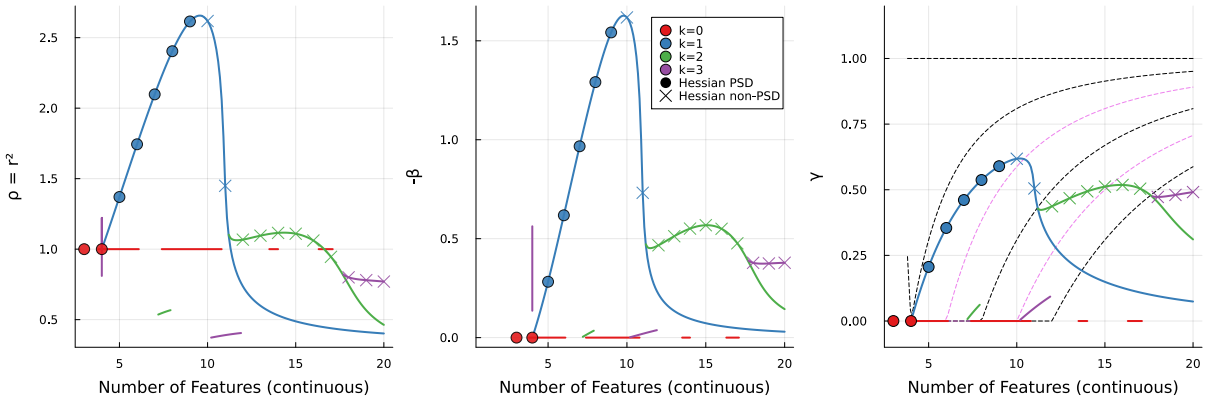


Figure 7: Plot of $\rho, -\beta, \gamma$ of the analytical continuous representation drawn as lines. Discrete solutions shown as points; solutions of the continuous representation shown as lines. Number of interfering neighbors $2k$ denoted by colors. Positive semidefinite (PSD, $H \geq 0$) Hessian solutions marked with circles. Solutions with non-PSD Hessian marked with crosses.

Notably, the stability of the solutions seems to switch at the maximum of the curve, after which there are only 2 stable solutions. It seems that for higher m, higher radii and lower biases are generally needed to stabilize the solutions.

Also notably, the maxima of the blue $k = 1$ and green $k = 2$ curves almost coincide with the mid-neighbor lines in the γ-plot.

We now investigate the loss of each of the solutions, and its comparison to other solutions. How much advantage does a higher vertex solution bring compared to an embedded lower vertex solution and how much does this increase in higher dimensions? Practically, with usual sampling and AdamW, we rarely see more than hexagonal structures.

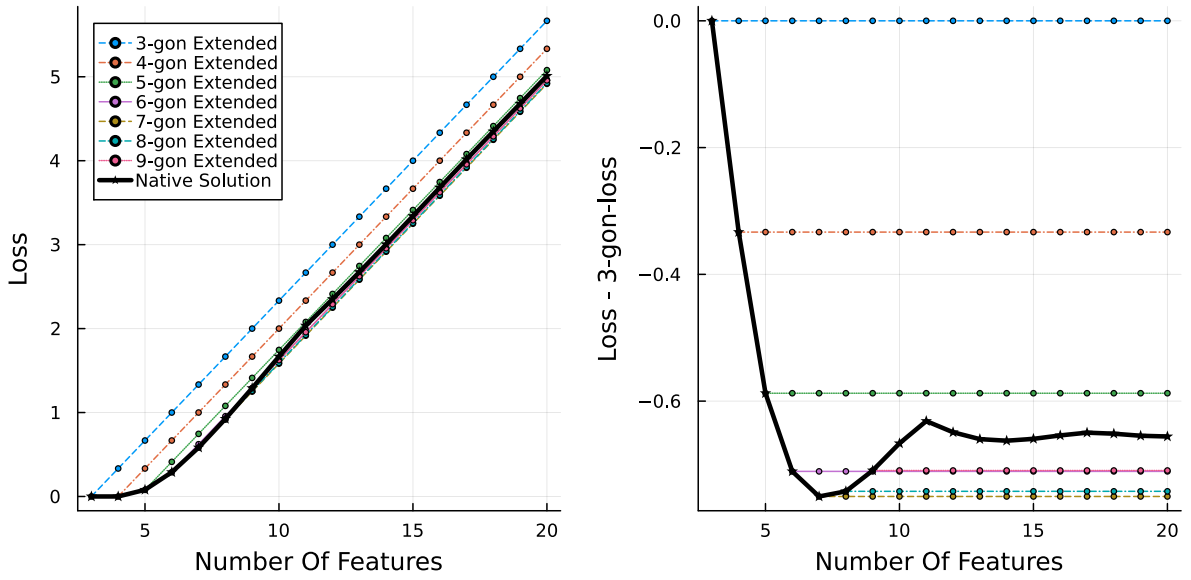


Figure 8: The loss of the polygon solution with exactly m vertices plotted against the loss of lower-vertex polygon solutions in higher feature numbers on the left; the loss difference to the 3-gon solution is shown on the right. We see that, starting at $m = 9$, there are polygon solutions with fewer than m vertices that have a lower loss than the polygon solution with exactly m vertices.

The progressive loss improvement over previous solutions decays rapidly: From the beginning until the hexagon, we see visible improvement over embedded solutions, while at $m = 9$, the octagon

solution is clearly better than the native 9- and 10-gon solutions. This explains why lower-order polygons are much more prevalent—the interference for higher n-gon solutions increases superlinearly for some time, while the loss of embedded solutions only increases linearly. However, the native solution seems to stabilize after 11, with approximately linear growth.

The diminishing returns from higher-order polygons explain why hexagons are the most complex structures we typically observe. Beyond 8 features, embedding lower-order polygons yields better loss than native solutions, with the return already strongly diminishing from 6 features onward.

We now investigate another aspect of the unstable polygon solutions: In which directions do they disintegrate when optimized? Because they are symmetric, and the loss is invariant under rotation, it is not obvious what this exact direction looks like. We plot the corresponding $\boldsymbol{W}$ components of this eigenvector of minimum eigenvalue of the Hessian.[6]

[6]For finding eigenvectors and eigenvalues of the symmetric Hessian, we use the `eigen` function of Julia's LinearAlgebra package. This calls LAPACK's `xSYEVD` algorithm, which uses a tridiagonal reduction followed by a divide and conquer eigensolve.

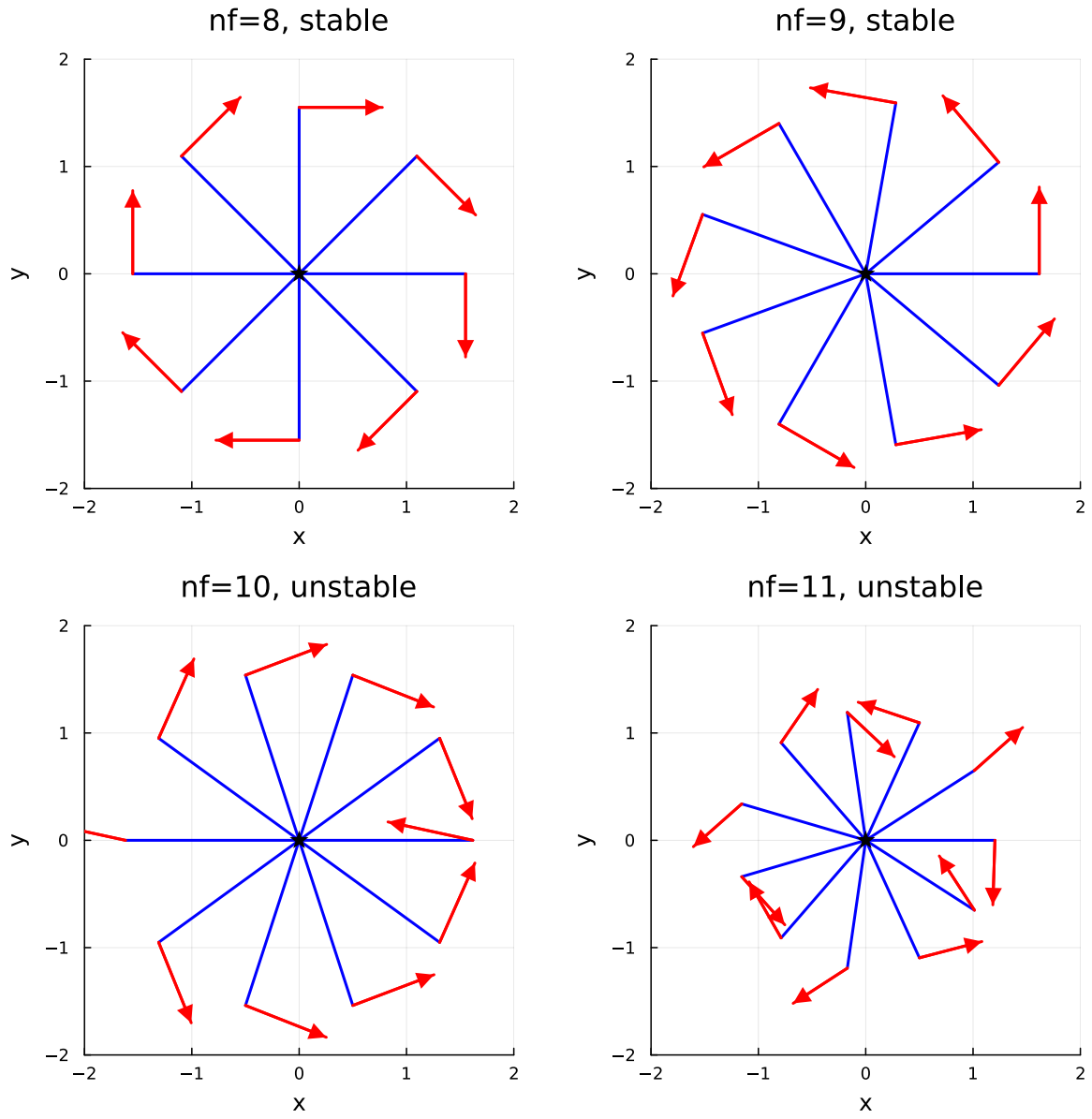


Figure 9: Descent directions of exemplar stable and unstable polygon solutions. Here, by descent direction, we mean the eigenvector of the smallest eigenvalue, which is 0 for stable solutions and negative for unstable solutions.

For the stable solutions, we see the expected pattern: The minimum eigenvalue is 0, and the corresponding eigenvectors correspond to rotation of the polygon in some direction. For the unstable solutions, the minimum eigenvalue is negative, and the direction twists apart two adjacent vertices, while some of the vertices are pulled inwards or outwards, corresponding to a complex twisting motion. In perturbation experiments, we see mostly a decay of the 10-gon into a regular 8-gon or 9-gon, which seems plausible given the observed pattern for $m = 10$.

3.3 Empirical Validation and Extensions

Having characterized superposition theoretically through loss analysis and polygon solutions, we now turn to empirical validation. We examine how robust the superposition phenomenon is to architectural choices and training procedures, and investigate the dynamics of how it emerges during optimization.

3.3.1 Measuring Feature Representation

Before conducting experiments, we need precise metrics to quantify feature representation. When reproducing Figure 2, we observe that, depending on different seeds, the formation of pentagons changes significantly. Moreover, it is not straightforward to measure feature representation: as we can see in Figure 2, there are different strategies of feature representation that can be learned. The model might learn the most important two features fully, but then learn nonzero smaller weight vectors for the other features; so, in a sense, the non-important features are 'half-represented'.

We introduce two complementary approaches to measure feature representation, which we validate empirically before applying them to study the robustness of superposition in Section 3.3.2:

Error-based feature representation. We compute the reconstruction error $\max_j (\mathcal{M}(e_i) - e_i)_j^2$ for each feature i. If this error is below a threshold ε, we count the feature as represented. We use the maximum over all output dimensions, which allows us to compare ε across different feature numbers. This metric is agnostic to the specific weight vector representation and could in principle be used for uncoupled and more complex models. However, if an important feature shares a significant portion with another feature (as in low-sparsity models in Figure 2), the metric may not count it as represented.

Weight-based feature representation. We compute the 2-norm of the weight column vector W^i of each feature i. If this norm is above a threshold min-norm, we count the feature as represented. This metric is specific to the coupled toy model, but it is more robust when features share components with other features.

To validate these metrics and determine appropriate threshold values, we plot the number of represented features against feature probability for various threshold settings.

For each individual feature probability $(1 - S)$, we train 16 randomly initialized models and compute how many features are represented by each metric. We plot average and standard deviation in Figure 10.

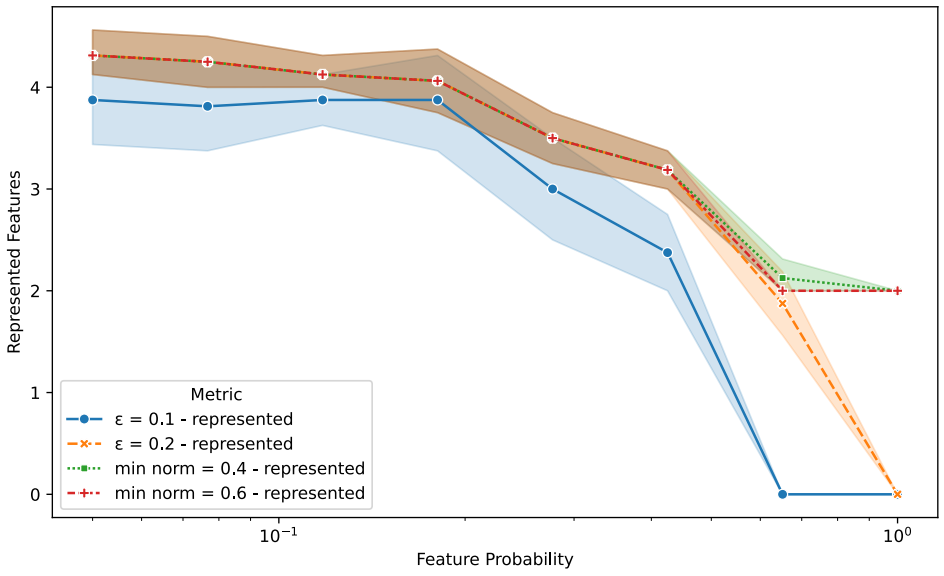


Figure 10: Approximate feature representation vs. sparsity plotted with standard deviation.

We see that the average number of represented features rises consistently as features become sparser. The two metrics agree closely in the low-feature-probability regime: the error-based metric

with $\varepsilon = 0.2$ aligns almost perfectly with the weight-based metric with min-norm $= 0.4$. We adopt these parameter settings as our standard metrics for subsequent experiments.

The disagreement in the high-feature-probability regime occurs because when features are likely to be active, the model learns non-zero positive biases for unrepresented features, leading to higher reconstruction error. The weight-based metric correctly identifies two represented features in this regime.

In subsequent experiments, we use the same hyperparameters as in Figure 10 for a "small" configuration. We also employ a "large" configuration with $m := 100$ features and $n_h := 10$ hidden dimensions, using a smaller importance decay of $I_i := 0.999^{i-1}$ to accommodate the increased feature count. The large model exhibits more regular behavior, allowing us to reduce the number of repetitions to 4 per sparsity level.

3.3.2 Robustness to Architectural Choices

We now examine whether the superposition phenomenon is robust to variations in model architecture. Specifically, we investigate different activation functions and the distinction between coupled versus uncoupled weight matrices.

Activation Functions

If superposition is a fundamental phenomenon in neural networks, it should occur with different activation function variants, at least those similar to ReLU. While the toy model reconstruction problem seems specifically designed to be optimally solvable with the ReLU activation function, one can transform other activation functions to a ReLU-like form using linear transformations. We thus expect superposition in general networks even for many non-ReLU activations, given that it robustly appears for many such variants. We focus on ReLU variants in this section.[7]

We train models with different variants of LeakyReLU, evaluating weight-based feature representation. (We avoid error-based metrics here since the reconstruction error fundamentally changes with the activation function.) LeakyReLU variants interpolate between ReLU and the identity function: $\text{LeakyReLU}(\alpha) = (1-\alpha)\,\text{ReLU} + \alpha\,\text{id}$, allowing us to investigate how the degree of non-linearity affects feature representation.

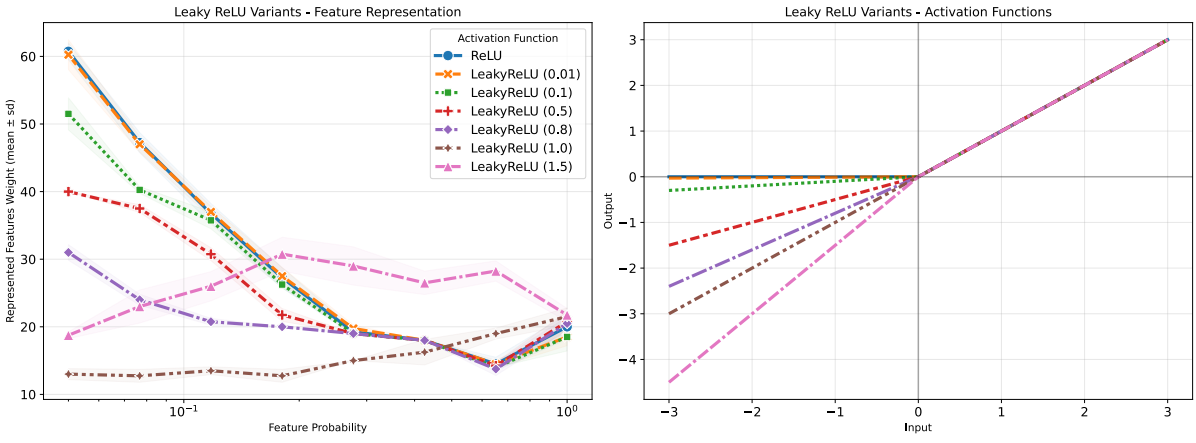


Figure 11: Approximate feature representation vs. sparsity for different Leaky ReLU variants, as measured by weight-based feature representation shown on the left. LeakyReLU variants used plotted on the right.

[7] The code for the activation function comparison can be found in `src/masterthesis/experiments/activation_function_experiments/activation_function_comparison.py`.

We observe a gradual decrease in the number of represented features as the LeakyReLU variant deviates from pure ReLU. The effect is most pronounced in the low-feature-probability regime, while in the high-feature-probability regime the variants behave similarly. Interestingly, the identity function LeakyReLU(1.0) leads to approximately $10 - 20$ weight-represented features—more than the number of hidden dimensions—suggesting that attractive local superposition solutions exist even in the linear case.

The outlier LeakyReLU(1.5) leads to a relatively high number of weight-represented features in the high-feature-probability regime but few in the low-feature-probability regime. To investigate whether this represents a genuinely better solution, we compare the losses across variants in Figure 12.

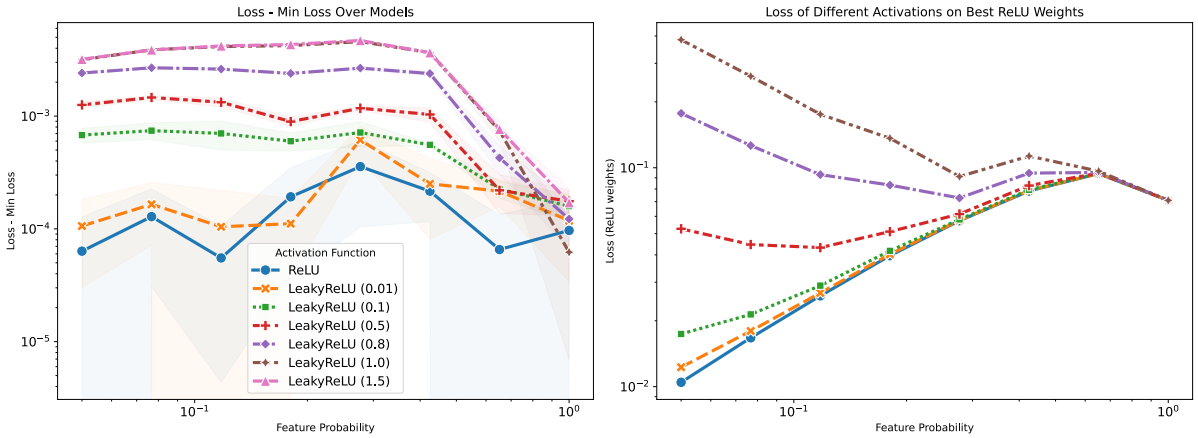


Figure 12: Left: We compare the Loss − Min Loss of the models trained with different LeakyReLU variants, where Min Loss is the minimum loss of any model for that feature probability. Right: We find the minimum-loss ReLU model and evaluate the loss for the respective LeakyReLU variant.

The LeakyReLU(1.5) variant achieves approximately the same loss as the linear case, confirming that its higher feature count is an optimization artifact rather than a better solution. For other variants, losses are evenly spaced on the log scale.

The behavior in the sparse regime aligns with our theoretical l_1-loss analysis from Section 3.1.2, specifically (3.2): replacing ReLU with LeakyReLU preserves negative interference, making superposition less beneficial. This effect is visible in the right plot of Figure 12, where loss increases for each LeakyReLU variant, most strongly in low-feature-probability regimes.

A secondary effect in Figure 11 is that LeakyReLU(0.01) sometimes achieves the highest average number of represented features, likely because it lacks a zero-gradient region, improving optimization reliability.

We now conduct an analogous comparison with the ELU activation function, another common ReLU variant. The ELU activation is defined as

$$\mathrm{ELU}(x) = \begin{cases} x < 0 : \alpha(\exp(x) - 1) \\ x \geq 0 : x \end{cases}.$$

It is differentiable if and only if $\alpha = 1$.

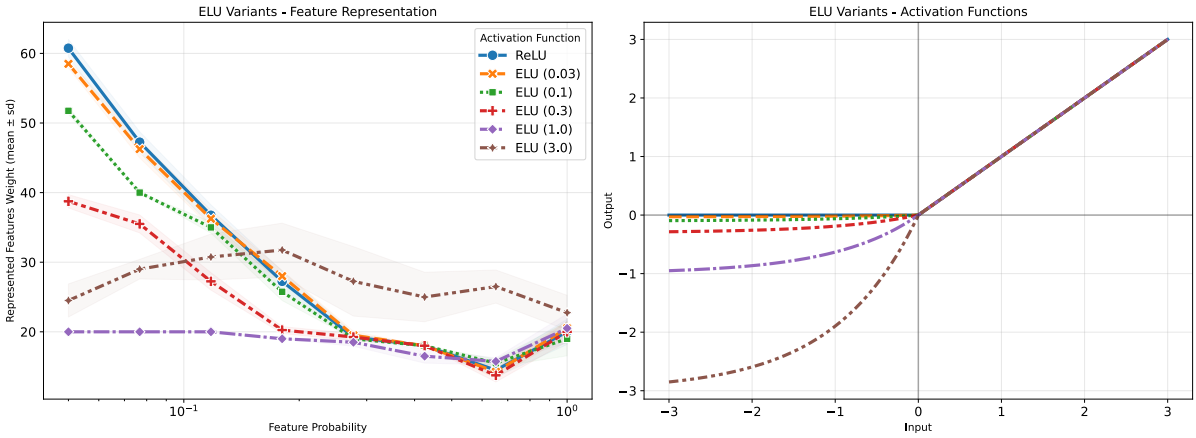


Figure 13: Approximate feature representation vs. feature probability for different ELU variants, as measured by weight-based feature representation shown on the left. ELU variants used plotted on the right.

The behavior closely parallels the LeakyReLU results: a gradual decrease in represented features with increasing α, and the same outlier pattern for large $\alpha = 3$.

Coupled vs Uncoupled Models

We now investigate the effect of weight coupling, which was a core design choice of Definition 3.2.[8] In the standard toy model, features are coupled through a single weight matrix W used for both encoding and decoding. We compare this to an uncoupled model with two distinct weight matrices:

$$\mathcal{M}(X) := \mathrm{ReLU}\big(\boldsymbol{W}_{\uparrow}\boldsymbol{W}_{\downarrow}X + b\big).$$

We compare three configurations using both small and large model settings: (1) the coupled model, (2) an uncoupled model with standard Kaiming uniform initialization, and (3) an uncoupled model with Xavier normal initialization where $\boldsymbol{W}_{\uparrow}$ is initialized as the transpose of $\boldsymbol{W}_{\downarrow}$.

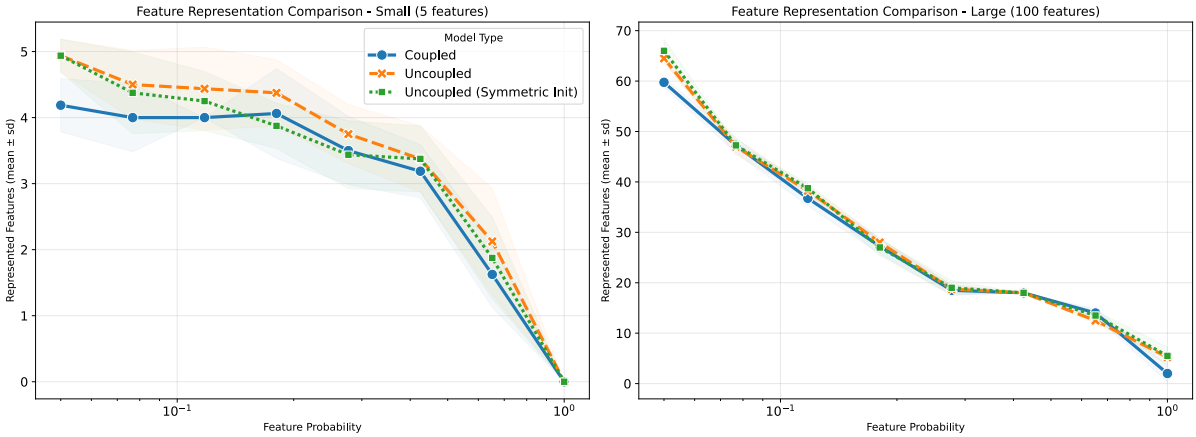


Figure 14: Approximate feature representation vs. feature probability for coupled and uncoupled models, as measured by error-based feature representation. The left plot is for the small experiment with $m = 5$, $n_h = 2$, the right plot is for the large experiment with $m = 100$, $n_h = 10$.

The uncoupled model slightly outperforms the coupled model in both configurations. In the small model, the uncoupled model with non-symmetric initialization performs best, suggesting either more reliable optimization or access to better solutions. In the large model, symmetric initialization

[8] The code for the coupled vs uncoupled model comparison can be found in `experiments/coupled_model_investigation/feat_repr_coupled_uncoupled_new.py`.

performs best, possibly because it provides a beneficial prior, though better solutions may exist that the optimizer fails to find with non-symmetric initialization.

To examine how coupling affects learning dynamics, we plot feature representation over training in Figure 15.

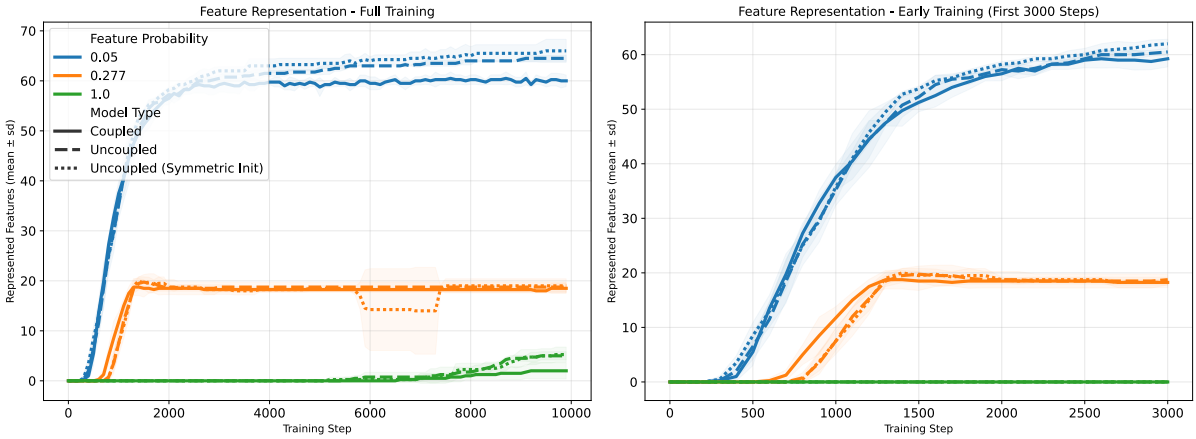


Figure 15: Approximate feature representation vs. feature probability over training for the large experiment with $m = 100$, $n_h = 10$. The left plot is for the coupled model, the right plot is for the uncoupled model with symmetric initialization.

The uncoupled model learns features slightly more slowly than the coupled model initially but eventually surpasses it. This is expected given the larger parameter count in the uncoupled architecture.

3.3.3 Training Dynamics

Having established the robustness of superposition to architectural variations in Section 3.3.2, we now investigate how it emerges during training and how sensitive it is to optimization hyperparameters. This complements our theoretical analysis of the loss landscape in Section 3.1.2.

Optimizer Parameters

We examine how sensitive superposition is to optimization hyperparameters.[9] We focus on AdamW, a variant of the Adam optimizer that decouples weight decay from the optimization process, commonly used in practice. We investigate the effects of momentum, learning rate, and weight decay.

Momentum. We first examine the effect of the momentum parameter β_1.

[9]The code for the optimizer comparison can be found in `src/masterthesis/experiments/optimizer_comparison/optimizer_comparison.py`.

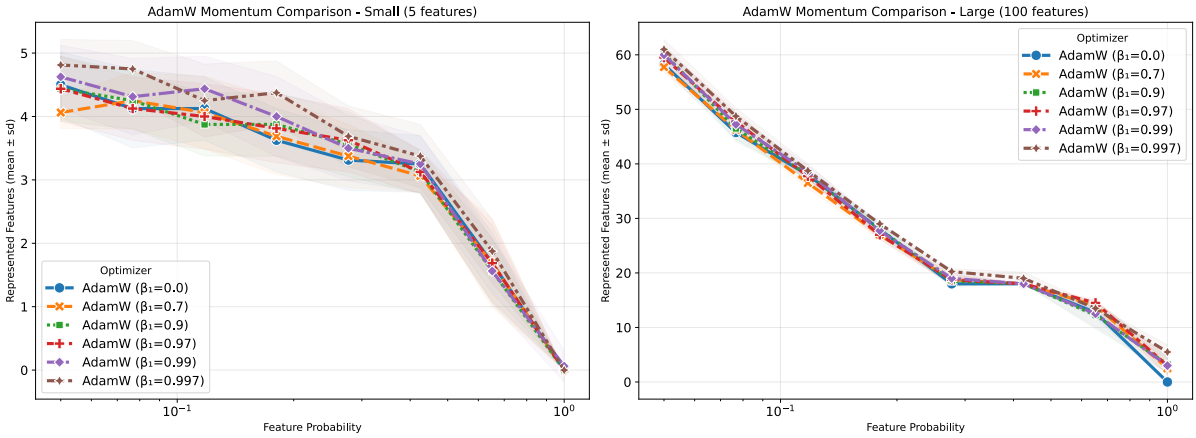


Figure 16: Approximate feature representation vs. feature probability for different AdamW momentum values, as measured by error-based feature representation. The left plot is for the small experiment with $m = 5$, $n_h = 2$, the right plot is for the large experiment with $m = 100$, $n_h = 10$.

There is a small but significant effect of momentum β_1 in the small experiment, where increasing momentum leads to higher feature representation. In the large experiment, the effect is smaller but still visible. This likely occurs because momentum smooths the large variation in batch gradients due to sparse sampling, an effect more pronounced in small experiments. In analogous SGD experiments, momentum enables feature learning entirely—without momentum, SGD fails to learn any features.

Learning rate. We now examine the effect of learning rate.

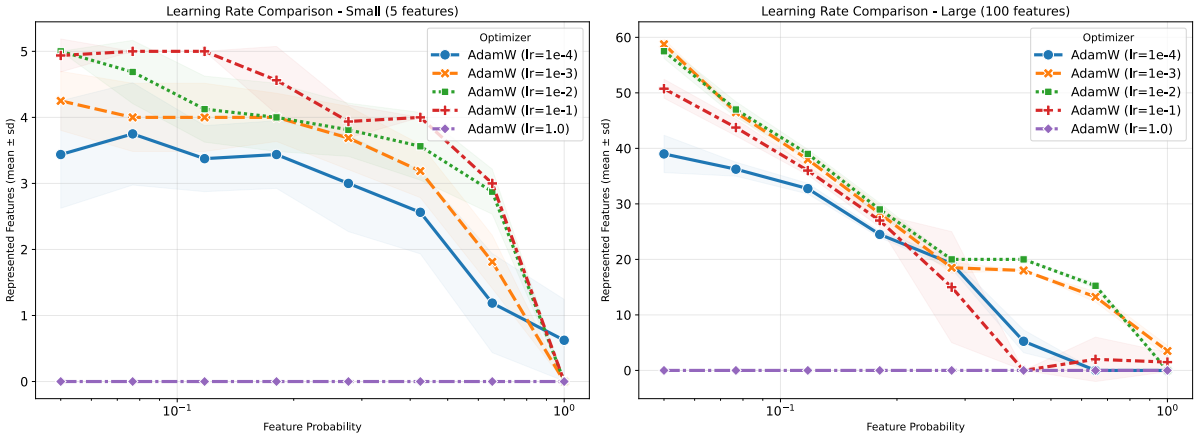


Figure 17: Approximate feature representation vs. feature probability for different learning rates, as measured by error-based feature representation.

Learning rate has a strong effect on feature representation. The small model benefits from a higher learning rate of 10^{-1}, whereas the large model benefits from lower learning rates of 10^{-2} or 10^{-3}.

Weight decay. The effects of weight decay appear modest, though we find that the small model benefits from a small weight decay of 10^{-2}.

Emergence over Training

We now investigate how feature representation emerges over training.[10] We train multiple models for each feature probability over 10000 steps, evaluating represented features every 100 steps. This

[10] The code for the emergence over training experiment can be found in `src/masterthesis/experiments/feat_repr_over_training.py`.

allows us to observe how the benefit-interference tradeoff, analyzed theoretically in Section 3.1.2, evolves during optimization.

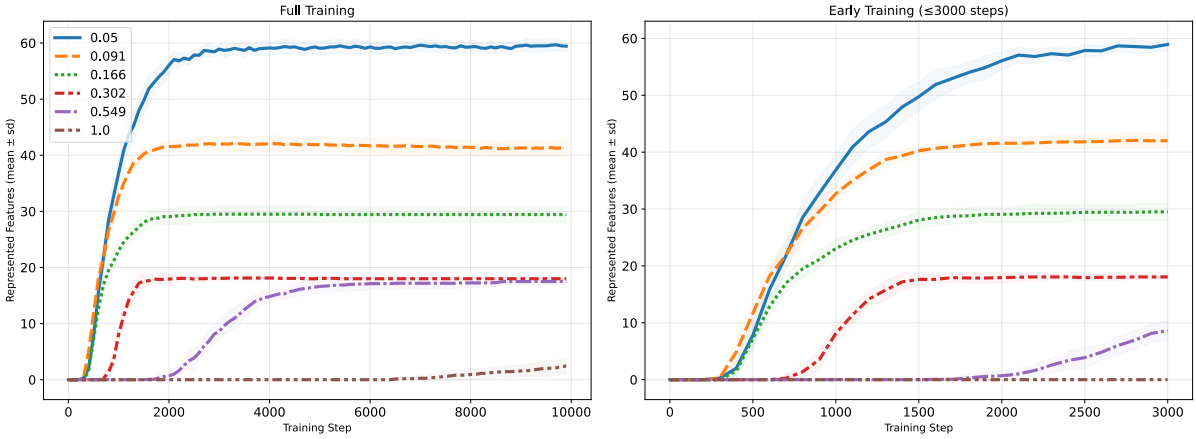


Figure 18: Approximate feature representation vs. feature probability over training for the large experiment with $m = 100$, $n_h = 10$. The left plot shows full training, the right the early training phase.

We repeat the experiment for the small model in Figure 19.

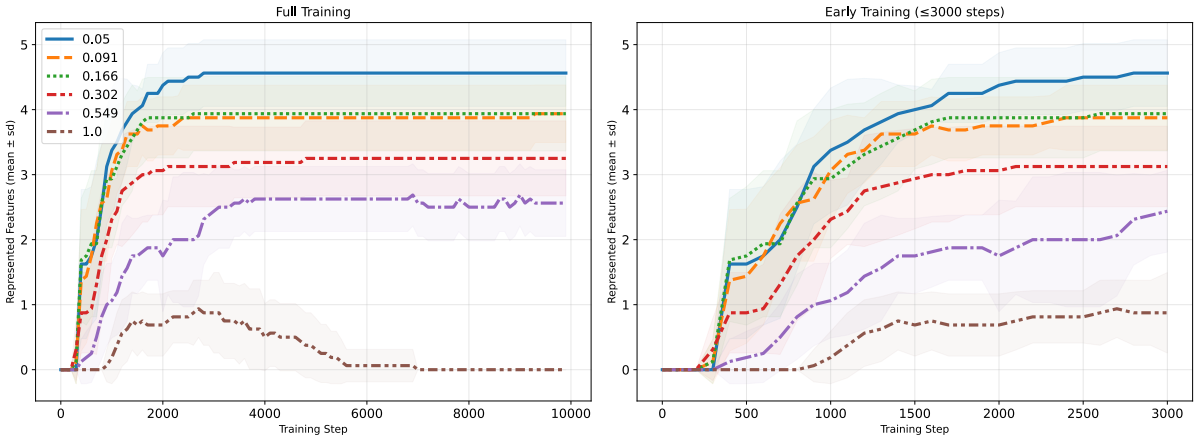


Figure 19: Approximate feature representation vs. feature probability over training for the small experiment with $m = 5$, $n_h = 2$. The left plot shows full training, the right the early training phase.

While most feature representation is achieved within the first 3000 iterations, the low-feature-probability models continue improving until the end of training. The pentagon configuration is learned primarily in the final 5000 iterations, consistent with the observation of multiple learning phases in [1]. We observe nearly discrete jumps in the number of represented features, suggesting phase transitions in the learning dynamics.

These phase transitions are most visible in the small model where all features occupy the same 2D plane; in the large model, multiple phase transitions likely occur but are less distinct.

3.4 Feature Geometry and Dimensionality

Having examined how superposition emerges and its robustness to architectural choices, we now investigate the geometric structure of learned representations more closely, particularly the average

[11] The code for the feature geometry stability experiment can be found in `src/masterthesis/experiments/feature_geometry_stability.py`.

dimensionality per feature.[11] This connects to our earlier theoretical characterization of regular polygon solutions in Section 3.2.1.

This section closely follows results from [1].

3.4.1 Sticky Points in Feature Geometry

The authors of [1] investigate $\frac{1}{n_h}\|\boldsymbol{W}\|_F$ as a proxy for the average number of hidden dimensions used per feature in the trained model. Their justification is that a represented feature corresponds approximately to a unit column vector in the weight matrix, so $\|W\|_F$ approximates the number of represented features.

Plotting this proxy over a range of sparsities for models with $n_h = 200$ hidden dimensions and $m = 100$ features, they observe a "sticky point" where the curve plateaus at $\frac{1}{2} = \frac{1}{n_h}\|\boldsymbol{W}\|_F$. To verify this result, we reproduce their experiments while training 8 models for each feature probability level.

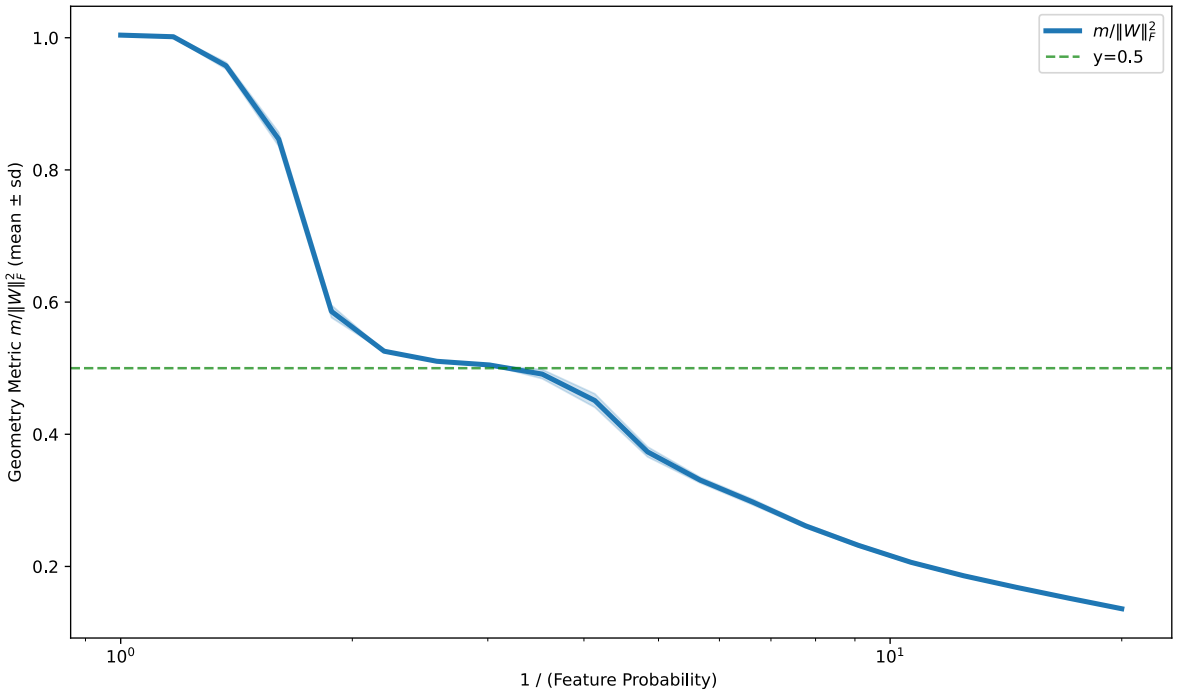


Figure 20: Proxy for average feature dimensions vs. feature probability with marked standard deviation.

The sticky point and general curve are remarkably stable with small standard deviation. This stability likely results from the medium number of hidden dimensions (200), contrasted with the high variability observed in 2D polygon optimization characterized in Section 3.2.2.

The explanation for the sticky point is that $\frac{1}{2}$ dimension per feature is precisely what is required to represent antipodal features (features pointing in opposite directions), and this antipodal configuration represents a preferred solution, connecting to the geometric structures analyzed in Section 3.2.1.

3.4.2 Defining Feature Dimension

Motivated by the sticky point results in Figure 20, we now formalize a notion of *feature dimension* based on how features are represented in the model.

Definition 3.20. (Feature dimension in the representation model [1]):
Given a set of feature vectors $F = \{f_i\}_i$, the *feature dimension* of f_j is defined as

$$D_j := \frac{\|f_j\|^2}{\sum_i \left(\hat{f}_i \cdot f_j\right)^2}.$$

This can be interpreted as the effective number of dimensions dedicated to representing the feature.

Example 3.21. (Feature dimension):
Consider two 'antipodal' features

$$F = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\} \subset \mathbb{R}^2.$$

The feature dimension for both features is $\frac{1}{1+1} = \frac{1}{2}$. Consider now

$$F = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -\frac{1}{2} \\ 0 \end{pmatrix} \right\}$$

Here, per definition, the feature dimensions change with the different norms of the features: For the first, we obtain $\frac{1}{1+\frac{1}{4}} = \frac{4}{5}$; for the second, $\frac{\frac{1}{4}}{1+\frac{1}{4}} = \frac{1}{5}$.

Example 3.22. (Feature dimension of a regular polygon):
Consider features distributed as a regular polygon in $\mathbb{R}^2$ as in Lemma 3.14, i.e., $z_i := (\cos(\Delta\varphi\ i), \sin(\Delta\varphi\ i))^T$, for $\Delta\varphi := \frac{2\pi}{m}$. Thus, the scalar product of two features is $f_i \cdot f_j = \cos(\Delta\varphi\ (i-j))$. Thus, the denominator of the feature dimension becomes

$$\sum_{j=0}^{m-1} \underbrace{\cos^2(\Delta\varphi\ j)}_{\frac{1}{2}(\cos(2\Delta\varphi\ j)+1)} = \frac{m}{2} + \underbrace{\sum_j \cos(2\Delta\varphi\ j)}_{=0 \text{ (x-coordinate of center of regular polygon)}}$$
$$= \frac{m}{2}$$

and we obtain $\frac{2}{m}$ as the feature dimension, so the features are split evenly across the two dimensions.

Note 3.23. (Properties of feature dimension):
The feature dimension is independent of the embedding dimension. Also, the feature dimension depends on the relative magnitudes of the feature vectors.

3.5 Summary

In this chapter, we investigated representational superposition through a minimal toy model following [1]. We formalized theoretical statements and then empirically evaluated the robustness of superposition in the toy model. We divided our analysis into three main parts:
(i) Theoretical analysis of the loss landscape and superposition conditions in Section 3.1.2.

Following [1], we defined the toy model as a coupled autoencoder architecture with ReLU activation. We analyzed the loss function by decomposing it according to the number of simultaneously active

features. We formalized and proved the statement from [1] that superposition is not advantageous in the linear case. However, with the ReLU non-linearity, one can decompose the asymptotically dominant l_1 loss term into benefit and interference components, revealing that superposition becomes advantageous when sparsity is sufficient to make interference costs worthwhile relative to the benefit of representing additional features.

(ii) Theoretical and numerical analysis of polygon solutions in Section 3.2.1

One interesting empirical observation from [1] is that regular polygons often emerge as solutions in the toy model. We analyzed these polygon solutions theoretically and numerically. We first analyzed the zero-bias loss, which appears qualitatively similar to the full loss but is more analytically tractable. We derived the gradient of the zero-bias loss and proved that regular polygons are stationary points of this loss, extending this to tegum products of polygons. For the general non-zero-bias loss, we proved that the loss gradient always lies tangent to the polygon manifold, enabling searching for stationary points on only the polygon manifold. We then used an analytical representation of the loss on the polygon manifold to compute solutions for different numbers of features. This analysis showed clear patterns: homogeneous polygon solutions in the non-trivial regime are essentially unique. Notably, starting at $m = 9$, the radius and bias of the solutions start to decline, which coincides with the solutions becoming unstable. We also observed that the number of interfering neighbors plays a crucial role in the stability of the solutions. Non-trivial stable polygon solutions only seem to exist for ≤ 2 interfering neighbors. We then compared the loss of polygon solutions to embedded lower-vertex polygon solutions, finding that starting at $m = 9$, embedded solutions have lower loss than native polygon solutions, which approximately coincides with the point where the solutions become unstable. The marginal loss returns explain why hexagons are the most complex structures typically observed in the toy model.

(iii) Empirical evaluation of superposition robustness and dynamics in Section 3.3.

We empirically investigated the behavior of the toy model under general variations of architecture and training procedure. We introduced two complementary metrics for measuring feature representation: error-based and weight-based, which agree closely in the sparse regime and for suitable choice of parameters. We found that superposition is robust to different activation functions, though the effect is weakened when negative interference is preserved (LeakyReLU variants). We also found that superposition occurs in both coupled and uncoupled weight architectures, which are closer to architectures in practice. Notably, with proper initialization, uncoupled models generally learn more features more reliably. Thus, it seems unlikely that actual architectures will learn coupled solutions similar to the polygon solutions analyzed theoretically. For variations in optimizer hyperparameters, we found that learning rate has the strongest effect on feature representation, while momentum and weight decay have small effects. We also investigated how superposition emerges over training. We found that most features are learned in the first half of training. Notably, the more complex highly superposed representations for high sparsity levels are learned later. In the small model, we observed indications of discrete phase transitions in feature representation, consistent with observations in [1].

While [1] provides several theoretical statements, which required more careful formalization, it seems to us that the empirical findings in [1] appear to be robust across most architectural and training variations, indicating that representational superposition is a more general phenomenon in neural networks.

Chapter 4

Computation in Superposition

In this chapter, we consider the question of computational superposition, i.e., how neural networks can compute with features stored in superposition. This is a generally more difficult problem than representational superposition, as the network has to not only store and retrieve features, but directly operate on them, without introducing too much interference. While the JL-lemma (Corollary 2.9) established an exponential estimate for the number of features representable in superposition, we will see that Boolean computation with features in superposition is more limited.

In this chapter, we are going to investigate random constructions of neural networks that can compute Boolean functions. This is a natural choice for a simple class of logical operations that can be composed to more complex computations. Having concrete models for layers that can compute in superposition is interesting for several purposes:

- It provides a proof of concept that efficient computation in superposition is possible at all.
- It provides concrete scaling results for the required network size depending on input sizes. These estimates are useful in computational methods, where one wants to extract features from trained networks.
- It helps to understand trade-offs between network size, error, and sparsity.

We first establish fundamental notions in Section 4.1, including concentration inequalities and Boolean circuits. Our main focus in the analysis is the efficient computation of And circuits in superposition, which is the foundation for computing arbitrary Boolean functions. We then give several variants of asymptotic representability of the UAnd circuit in Section 4.2.1, which includes modeling interference errors and analyzing the required width for single-layer networks. This is followed by analysis of the representability of UAnd in i.i.d. Gaussian-distributed weights in Section 4.2.2, which illustrates that the UAnd function can be represented in superposition by very general random constructions. We then extend to N-ary And circuits in Section 4.2.3 and quote results for multi-layer Boolean circuits in Section 4.3. Finally, we provide explicit estimates for the error of the Bernoulli and general i.i.d. constructions in Section 4.4. We conclude by showing empirical results comparing errors of different constructions and trained networks in Section 4.4.2.

We generally follow the results of [2]. The original paper provides short proof sketches for some of the statements we quote and contains full proofs only for a related—but different—setup in the appendix, which assumes polynomial relation of width and number of features and does not directly use the Bernoulli constructions that they use in their main text proof sketches. We build and prove precise statements related to their main text statements and upgrade statements to o.n.p. in width, where the width scales exponentially in the number of features. For this, we take different approaches in several proofs, referencing Gauss concentration and Chernoff bounds explicitly and fundamentally build on the Bernoulli constructions.

4.1 Fundamentals

We build on fundamental notions from computational complexity theory and concentration inequalities. Probabilistic tools and concentration inequalities are natural tools for analyzing neural networks behavior—we are not mainly interested in what neural networks in principle exist but also whether trained networks are likely to emulate similar behavior. The notion of negligible probability

conveniently allows qualitative reasoning for average-case behavior which often does not require tracking precise upper bounds.

For a list of notation, see Appendix A.1.

4.1.1 Negligible Probability

Working directly with explicit probability bounds can be cumbersome. Many of the bounds we encounter decay faster than any polynomial and can practically be ignored for large numbers. The notion of negligible probability captures this intuition.

Definition 4.1. (Negligible function):
A sequence $(p_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{\geq 0}$ is negligible if it vanishes faster than any inverse polynomial function. I.e., we have

$$\forall c \in \mathbb{N} : p_n \in \mathcal{O}(n^{-c}),$$

where $\mathcal{O}$ is w.r.t. $n \to \infty$.

Example 4.2. (Negligible functions):
A straightforward example of negligible function are exponential functions, e.g., $\exp(-n^c)$ is negligible for any $c > 0$. A class of negligible functions we will also encounter are functions of the form $\exp(-\log(n)^\alpha)$ for $\alpha > 1$.

Definition 4.3. (Negligible probability):
A sequence of events $(\mathcal{A}_n)_{n=1}^{\infty}$ holds with negligible probability if $(\mathbb{P}(\mathcal{A}_n))_n$ is negligible. Similarly, a sequence of events holds outside negligible probability (abbreviated o.n.p.) if $(1 - \mathbb{P}(\mathcal{A}_n))_n$ is negligible.

This notion allows qualitative reasoning about negligible events. For example, if polynomially many events in n hold outside negligible probability, then their intersection does so as well.

Lemma 4.4. (Intersection of polynomially many o.n.p. events):
Let $(\mathcal{A}_i^n)_{i=1}^{m_n}$ be a collection of polynomially many events for each $n \in \mathbb{N}$ that are true outside negligible probability, i.e., $m_n \in \mathcal{O}(n^c)$ for some $c > 0$ and $\mathbb{P}(\mathbb{C}(\mathcal{A}_i^n))$ is negligible. Then, the intersection of all events is also true outside of negligible probability. I.e.,

$$\mathbb{P}\left(\mathbb{C}\left(\bigcap_{i=1}^{m_n} \mathcal{A}_i^n\right)\right) \in \mathcal{O}\left(n^{-c'}\right) \qquad \forall c' \in \mathbb{N}.$$

Proof.
Using the union bound, and that the sum of polynomially many negligible functions is also negligible, we directly obtain the result. I.e.,

$$\begin{aligned}\mathbb{P}\left(\mathbb{C}\left(\bigcap_{i=1}^{m_n}\mathcal{A}_i^n\right)\right) &= \mathbb{P}\left(\bigcup_{i=1}^{m_n}\mathbb{C}(\mathcal{A}_i^n)\right)\\ &\leq \sum_{i=1}^{m_n}\mathbb{P}(\mathbb{C}(\mathcal{A}_i^n))\\ &\leq m_n \cdot \max_i \mathbb{P}(\mathbb{C}(\mathcal{A}_i^n)),\end{aligned}$$

which is negligible, as it is the product of a negligible function with a polynomial.

□

For reasoning with probability bounds, we will sometimes use an adapted Landau notation that includes logarithmic factors.

Definition 4.5. (Polylogarithmic Landau notation):
We write $f(n) \in \tilde{\mathcal{O}}(g(n))$ if there exists a constant $c \in \mathbb{R}$ such that $f(n) \in \mathcal{O}(g(n)\log(n)^c)$. Analogously, we write $g(n) \in \tilde{\Omega}(f(n))$ if there exists a constant $c \in \mathbb{R}$ such that $g(n) \in \Omega(f(n)\log(n)^c)$.

Note 4.6. (Reasoning with intervals and bounds):
In this chapter, we often reason about random variables depending on some asymptotic parameter $m \in \mathbb{N}$ being contained in some bounds outside of negligible probability. For this it is convenient to combine interval notation with Landau notation and probabilistic statements.

For this we define the "function interval" $[x(m) \pm \Delta\mathrm{x}(m)]$ as the set of functions

$$[x \pm \Delta\mathrm{x}] := \{f \mid \forall m \in \mathbb{N} : f(m) \in [x(m) - \Delta\mathrm{x}(m), x(m) + \Delta\mathrm{x}(m)]\},$$

where $f, x, \Delta\mathrm{x}$ share the same domain and $[x(m) - \Delta\mathrm{x}(m), x(m) + \Delta\mathrm{x}(m)] \subseteq \mathbb{R}$ is the usual real interval.

Extending this notation to random variables, we write statements of the form

$$X \in [\mu \pm \mathcal{O}(f)] \qquad \text{o.n.p.},$$

where X, μ are random variables depending on some asymptotic parameter m and f is a function of m. This means that there exists some function $f'(m) \in \mathcal{O}(f(m))$ such that $\mathbb{P}(X(m) \notin [\mu(m) \pm f'(m)])$ is negligible.

We sometimes abuse notation and write $X(m) \in [\mu(m) \pm \mathcal{O}(f(m))]$ o.n.p. to denote the same statement.

Also we sometimes write inclusion statements like

$$[\mu \pm \mathcal{O}(f)] \subseteq \mathcal{O}(g).$$

This means that if some function $h(m)$ is in $[\mu \pm f']$ for some $f' \in \mathcal{O}(f)$, then this function is also in $\mathcal{O}(g)$.

4.1.2 Concentration Inequalities

Concentration inequalities provide bounds on the probability of how far a random variable deviates from some value, usually its expectation. We go through several foundational inequalities that we will use in the following.

Probably the most well-known fundamental concentration inequalities are the Markov and Chebyshev inequalities. However, these would only give polynomial probability bounds in our setting and are thus too weak to give negligible probability bounds directly.

In [2], the authors only quote the Bernstein inequality and base most of their proofs in the appendix on it. However, this is not sufficient to prove several of the statements in their main text, which require the Chernoff inequality. We also quote the Gaussian concentration inequality for Lipschitz functions of Gaussians, which we use to bound Gaussian interference terms later.

We first consider the classical Chernoff bounds, which provide strong concentration bounds for sums of independent Bernoulli random variables.

Proposition 4.7. (Chernoff bound [40, Cor. 4.6]):
Let $X = \sum_{i=1}^{n} X_i$ be a sum of independent Bernoulli random variables $X_i \sim \texttt{Ber}(p)$, i.e., where each $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then, for all $0 < \delta < 1$,

$$\mathbb{P}(|X - \mu| \geq \delta\mu) \leq 2\exp\left(-\frac{1}{3}\mu\delta^2\right).$$

The Chernoff bound gives strong bounds, however, is only applicable to sums of Bernoulli random variables. Hoeffding's inequality is a more general concentration inequality that generalizes the Chernoff bounds to arbitrary bounded independent random variables.

Theorem 4.8. (Hoeffding's Inequality [41, Thm. 2.8]):
Let $\Delta\mathrm{X}_1, \ldots, \Delta\mathrm{X}_n$ be independent random variables such that X_i takes its values in $[a_i, b_i]$ almost surely for all $i \leq n$. Let

$$X_n = \sum_{i=1}^{n} (\Delta\mathrm{X}_i - \mathbb{E}[X_i]).$$

Then for every $\Delta\mathrm{x} > 0$,

$$\mathbb{P}(X_n \geq \Delta\mathrm{x}) \leq \exp\left(-2\frac{\Delta\mathrm{x}^2}{\sum_{i=1}^{n} (b_i - a_i)^2}\right).$$

Note 4.9.
Equivalently, if one has the symmetric boundedness condition $|\Delta\mathrm{X}_i| \leq c_i$, then one can set $a_i = -c_i$ and $b_i = c_i$. This gives the convenient form

$$\mathbb{P}(X_n \geq \Delta\mathrm{x}) \leq \exp\left(-\frac{1}{2}\frac{\Delta\mathrm{x}^2}{\sum_{i=1}^{n} c_i^2}\right).$$

This is the form we will usually apply.

We can now state practical o.n.p. bounds directly from Hoeffding's inequality for our setting of interest. In [2, Thm. 29], the authors state a 'coarse Bernstein bound' for martingales or sums of i.i.d. variables, leaving out the proof. As we do not require the notion of Martingales, we give a reduced version based directly on Hoeffding's inequality.

Proposition 4.10. (Coarse Bernstein bound):
Let $\Delta X_1, ..., \Delta X_n$ be independent bounded random variables, i.e., $|\Delta X_i| \leq C$ for some constant $C > 0$. Denote the sum as $X_n := \sum_i \Delta X_i$ with expectation μ_n. Then, the deviation of the sum is bounded as

$$X_n \in \left[\mu_n \pm \log(n)\sqrt{n}\right] \subseteq \left[\mu_n \pm \tilde{\mathcal{O}}(\sqrt{n})\right]$$

outside of negligible probability.

Proof.
We apply Hoeffding's inequality with $c_i = C$ and $\Delta x = \log(n)\sqrt{n}$ to obtain

$$\begin{aligned}\mathbb{P}\left(|X_n - \mu_n| \geq \log(n)\sqrt{n}\right) &\leq 2\exp\left(-\frac{1}{2}\frac{\log(n)^2 n}{nC^2}\right) \\ &= 2\exp\left(-\frac{1}{2}\frac{\log(n)^2}{C^2}\right),\end{aligned}$$

which is negligible, as any function of the form $\exp(-\log(n)^\alpha)$ with $\alpha > 1$.

□

Example 4.11. (Number of ones in random bit vectors [2, c.f. Ex. 1]):
Let $X \in \{0,1\}^n$ be a random bit vector where each entry is 1 with probability $\frac{1}{2}$. Then, the number of ones in X is $\|X\|_0 \in \left[\frac{n}{2} \pm \log(n)\sqrt{n}\right]$.

Finally, we will need another inequality, which allows us to bound functions of Gaussians, such as ReLUs. Gaussians are not directly covered by Hoeffding's inequality, as they are unbounded. Notably, another general approach to bound functions of Gaussians, which is used in [2], is to first bound the Gaussians outside negligible probability and then apply Hoeffding's inequality.

Theorem 4.12. (Concentration for Lipschitz functions of Gaussians [42, Thm. 2.1.1]):
Let $X \sim \mathcal{N}(0, \mathbf{1}_n)$ be a standard normal random variable and let $f : \mathbb{R}^n \to \mathbb{R}$ be a Lipschitz function with Lipschitz constant $L > 0$. Then, for any $\Delta x > 0$,

$$\mathbb{P}(|f(X) - \mathbb{E}[f(X)]| \geq \Delta x) \leq 2\exp\left(-\frac{\Delta x^2}{4L^2}\right).$$

4.1.3 Probability Theory

We are going to use some notions from probability theory. The following are convenient shorthands for common cases.

Definition 4.13. (Rademacher random variable [43, p. 42]):
A random variable that takes values in $\{-1, 1\}$ is called a Rademacher random variable. We write $R \sim \texttt{Rad}(p)$ for such a variable, which is 1 with probability p and -1 with probability $1 - p$. If not specified, we set $p = \frac{1}{2}$.

In this thesis, we only consider Rademacher random variables that are symmetric, i.e., that take values -1 and 1 with equal probability $\frac{1}{2}$. This is the most important special case, which justifies the additional name.

Definition 4.14. (Half-normal random variable):
A random variable is half-normal if it can be written as the absolute value of a standard normal random variable. I.e., $X \sim \mathcal{N}_+(\mu, \sigma^2)$ iff $X = |Y|$ for some $Y \sim \mathcal{N}(\mu, \sigma^2)$.

Example 4.15. (Gaussian separation):
A zero-mean Gaussian random variable $X \sim \mathcal{N}(0, \sigma^2)$ can be written as the product of a Rademacher random variable $R \sim \texttt{Rad}$ and a half-normal random variable $Y \sim \mathcal{N}_+(0, \sigma^2)$, i.e., $X = RY$ with R and Y independent.

4.1.4 Boolean Circuits

In order to investigate Boolean circuits, we give the notions of Boolean gates and Boolean circuits. Generally, a (neural network) circuit is understood to be a modular piece of the computational structure of a neural network analogous to an algorithm in computer science. Mathematically, we here simply define Boolean circuits as compositions of Boolean gates, which are 2-ary Boolean functions. The following definition is adapted and extended from [2, p. 3 'Sparse Boolean Circuits'].

Definition 4.16. (Boolean circuit, Boolean gate, and sparsity):
A Boolean gate of fan-in (or arity) k is a Boolean function with k inputs, i.e. a function $g : \{0,1\}^k \to \{0,1\}$.

We call a Boolean circuit layer $\mathfrak{C}_l$ a Boolean function $\mathfrak{C}_l : \{0,1\}^{d_l} \to \{0,1\}^{d_{l+1}}$ which is a collection of Boolean gates of maximum fan-in 2, i.e., for each $i \in 1{:}d_{l+1}$, $(\mathfrak{C}_l)_i$ can be written as a Boolean function of two of the inputs.

A Boolean circuit of depth L is a Boolean function $\mathfrak{C} : \{0,1\}^{d_0} \to \{0,1\}^{d_L}$ that can be written as a composition of Boolean circuit layers $(\mathfrak{C}_l)_l$, i.e., $\mathfrak{C} = \mathfrak{C}_L \circ \cdots \circ \mathfrak{C}_1$. We call d_L the width of the circuit.

Definition 4.17. (Universal And circuit [2, Def. 4]):
The universal 2-AND circuit $\mathfrak{C}_{\text{UAnd}} = \mathfrak{C}^2_{\text{UAnd}}$ of a Boolean vector $x \in \{0,1\}^m$ is defined as the collection of all And gates on the m inputs, i.e.,

$$\mathfrak{C}_{\text{UAnd}}(x)_{k,l} = x_k \wedge x_l$$

for any indices $k, l \in 1{:}m$ with $k < l$. Analogously, the universal n-AND circuit $\mathfrak{C}^n_{\text{UAnd}}$ is defined as the collection of all possible n-ary And gates on the m inputs. I.e.,

$$\mathfrak{C}^n_{\text{UAnd}}(x)_I = \bigwedge_{i \in I} x_i$$

for any index set $I \subseteq 1{:}m$ with size $|I| = n$.

Example 4.18. (n-ary And):
Consider the n-ary And circuit $\mathfrak{C}(x) := x_1 \wedge \cdots \wedge x_n$. This circuit has depth $L = \lceil \log_2(n) \rceil$, as it can be computed by a binary tree of And gates. I.e., we define Boolean circuit layers $\mathfrak{C}_l$ as

$$\mathfrak{C}_1(x) := (x_1 \wedge x_2, x_3 \wedge x_4, ...),$$
$$\mathfrak{C}_2(x) := (\mathfrak{C}_1(x)_1 \wedge \mathfrak{C}_1(x)_2, \mathfrak{C}_1(x)_3 \wedge \mathfrak{C}_1(x)_4, ...),$$

and so forth, until we reach a single output after L layers.

Importantly, one can represent any Boolean function as a linear combination of ANDs, which allows us to focus on And circuits in the asymptotic analysis. The And circuit is a natural representation for sparse Boolean circuits because it preserves sparsity.

Example 4.19. (Linear representations of simple Boolean functions):
Consider Boolean variables $x_1, x_2, x_3 \in \{0, 1\}$. The 2-Or and 3-Or functions can be represented as

$$x_1 \vee x_2 = x_1 + x_2 - x_1 \wedge x_2,$$
$$x_1 \vee x_2 \vee x_3 = x_1 + x_2 + x_3 - (x_1 \wedge x_2) - (x_1 \wedge x_3) - (x_2 \wedge x_3) + (x_1 \wedge x_2 \wedge x_3).$$

Similarly, the XOr function can be represented as

$$x_1 \oplus x_2 = x_1 + x_2 - 2(x_1 \wedge x_2).$$

Proposition 4.20. (Linear representability of arbitrary Boolean functions using Ands):
Let $f : \{0, 1\}^a \to \{0, 1\}$ be an arbitrary Boolean function. Then, f can be written as a linear combination of ANDs of the input variables, i.e.,

$$f(x) = \sum_{I \subseteq 1:a} c_I \bigwedge_{i \in I} x_i$$

for some coefficients $c_I \in \mathbb{Z}$.

Proof.
This follows from the fact that the 2^a Boolean And functions are independent (Lemma 1.3) and thus form a basis of the vector space of all Boolean-to-real functions $(\{0, 1\}^a \to \mathbb{R}) \triangleq \mathbb{R}^{2^a}$.

□

4.2 Boolean Computation in Superposition

In this section, we present the main theoretical results on representability of Boolean circuits. We start by investigating different representations of And circuits, gradually building up representations that operate on superposed inputs. Afterwards, we consider Gaussian weights and give statements building up to arbitrary Boolean circuits. The common theme here is that we give randomly constructed networks and analyze the relation of error, width and other parameters asymptotically.

Note 4.21. (Parameter overview and asymptotic relations):

- $m \in \mathbb{N}$: Boolean input length, i.e., the length of the input vector $x \in \mathbb{R}^m$, i.e., number of input features. This is our primary asymptotic parameter.
  - Specifically, our o.n.p. statements are w.r.t. m.
- $d(m) \in \mathbb{N}$: Layer width. The width is the number of outputs in a layer. Usually, the width $d(m)$ is assumed to be increasing in m.
- $\varepsilon = \varepsilon(m), \varepsilon_{\text{in}}, \varepsilon_{\text{out}}$: Error parameters. These denote some maximum deviation in input or output.
  - These are usually non-increasing in m.
  - When having multiple error parameters, we usually use $\varepsilon = \varepsilon(m)$ as target error parameter, for which we want to achieve that the actual error of the construction ε_{out} satisfies $\varepsilon_{\text{out}} \leq \varepsilon$.
- $s \in \mathbb{N}$: Sparsity parameter. This denotes the number of active features in the input. In our setup, we consider s to be constant.
  - In more general setups, we can allow s to grow with d.

We usually keep constant parameters explicitly in the Landau notation, which is useful to see the approximate scaling behavior.

Note 4.22. (Contribution):
We follow the main text framework of [2], with Bernoulli and Gaussian constructions and the asymptotic statements. However, we rebuild and prove variants of their main text statements. Specifically:

- The authors only give proof sketches for the statements in their main text. They provide more complete proofs for a similar, but different setup in their appendix, which are significantly more technical. Their appendix proofs deviate significantly from the main text proof ideas.
- The proof sketches in the main text often give logarithmic width scaling in m, while their appendix generally requires polynomial width scaling. This leads to the weak Bernstein bound being insufficient. We sometimes use additional concentration inequalities.
  - Specifically, we rely on the stronger Chernoff bound in our analysis (see, e.g., Corollary 4.29). Notably, the Chernoff inequality is not always sufficient, as expectations sometimes decay. We thus also add Lemma 4.36 to cover such cases. This also leads to different bounds being used for different width scaling regimes: For logarithmic width scaling, we can use Chernoff bounds directly, while for the polynomial width scaling in case of superposed inputs, we obtain a decreasing expectation and use Lemma 4.36. To accommodate this, we split the theorems into abstract propositions and apply them in different regimes. (e.g., Proposition 4.26 is the general statement, where Corollary 4.29 and Corollary 4.35 are applications to different regimes.)
  - Also, in case of non-constant ε and logarithmic width scaling, their definition of the Bernoulli weight probability of $\log(m)^2 d^{-\frac{1}{2}}$ breaks down, as it might rise for sufficiently quickly growing ε. Thus, we adapt the probability parameter to a definition based on ε and d in Corollary 4.29.
- The first constructions in the main text use Bernoulli initializations with a probability parameter p, while the appendix constructions use a more complicated $\{+1, -1, 0\}$ random initialization depending on graphs without an explicit probability parameter.
  - We build the full framework around the Bernoulli initialization case, as in their main text.
  - Notably, we find that the Bernoulli construction suggested by their main text proof sketch is biased and thus leads to worse noise behavior and larger width growth, $\tilde{\mathcal{O}}\left(m^{\frac{2}{3}}\right)$ vs. $\tilde{\mathcal{O}}\left(m^{\frac{1}{2}}\right)$, than in [2, Thm. 2]. See Corollary 4.35 for the statement and Note 4.54 for discussion. Notably, their appendix construction does not have this bias and uses a symmetric weight distribution.
  - We give concrete error expectation and variance estimates for the Bernoulli and general i.i.d. constructions in Appendix A.4, which allow giving precise error bounds. We also compare different constructions empirically in Section 4.4.2.

4.2.1 Computing UAnd in Superposition

As any Boolean function can be represented as a linear combination of ANDs, efficiently computing the universal And circuit $\mathfrak{C}_{\text{UAnd}}$ can be a foundation for computing arbitrary Boolean functions in superposition. Basing sparse computation on ANDs can be seen as a natural choice, as ANDs preserve sparsity of the inputs.

The fundamental idea is to trade off width against error and leverage averaging to reduce interference effects. This allows the layer width to be only polylogarithmic in the input dimension. This is significantly better than the naive approach of directly computing all ANDs, which requires a width of $d = \binom{m}{2} \in \Theta(m^2)$.

More concretely, the idea is to compute 'partial ANDs' of many random collections of variables. From this set of partial ANDs, one can then read off the And of any two variables with low error by

averaging over all partial ANDs that contain both variables. The 'partial And' operation is naturally implemented by ReLU neurons, which activate if at least two of their inputs are active.

We first define the Bernoulli UAnd network construction and the read-off vectors for reading out arbitrary ANDs from the network output.

Definition 4.23. (Bernoulli UAnd network, Γ-sets, read-off vector):
We define the n-ary Bernoulli UAnd network as a one-layer ReLU network $\mathcal{M} : \mathbb{R}^m \to \mathbb{R}^d$ with weight matrix $\mathbf{W}_{\texttt{Ber}} \in \{0,1\}^{d \times m}$ and bias $b := -(n-1)$, where each entry of $\mathbf{W}_{\texttt{Ber}}$ is independently drawn from a Bernoulli distribution with probability $0 < p < 1$, i.e.,

$$(\mathbf{W}_{\texttt{Ber}})_{i,j} \sim \texttt{Ber}(p) \qquad \forall i \in 1{:}d, j \in 1{:}m.$$

The one-layer network is defined as $\mathcal{M}(x) := \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x + b)$ for $x \in \mathbb{R}^m$.

For distinct feature indices $\{k_1, k_2, ..., k_K\} \subseteq \{1, ..., m\}$, we define the sets

$$\Gamma_K(k_1, k_2, ..., k_K) := \left\{ i \in 1{:}d \mid (\mathbf{W}_{\texttt{Ber}})_{i,k_j} = 1 \qquad \forall j \in 1{:}K \right\}.$$

We define the read-off vector for $x_{k_1} \wedge \cdots \wedge x_{k_n}$ as

$$r^{k_1,\ldots,k_n} := \frac{1}{|\Gamma_n(k_1, ..., k_n)|} 1_{\Gamma_n(k_1,\ldots,k_n)}$$

where 1_{Γ_n} is the indicator vector for the set $\Gamma_n(k_1, ..., k_n)$, which is 1 at index i if $i \in \Gamma_n(k_1, ..., k_n)$ and 0 otherwise.

If we do not specify n, we refer to the $n = 2$ case by default.

Example 4.24. (Reading off arbitrary ANDs from partial ANDs):
As an example for how one can leverage averaging over partial ANDs to compute arbitrary ANDs in superposition, consider $m = 8$ features and a width of 5. Let the weight matrix $\mathbf{W}_{\texttt{Ber}}$ be

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

In the theorem, the width for this input size would be significantly larger than practical for illustration, but asymptotically the width rises slowly with the input size.

We define a 1-layer network as $\mathcal{M}(x) := \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}}x + b)$ with bias $b := -1$. Let us consider the And of the first and third input, i.e., $x_1 \wedge x_3$. The neurons that activate for this And are the 1st and 5th neuron. We collect these in the set $\Gamma_2(1,3) := \{1,5\}$. We are going to use these sets in the proofs to estimate errors and construct read-offs.

In order to read-off the And of x_1 and x_3 from the network output, we average over the outputs of these neurons. We can conveniently do this using a scalar product with a read-off vector,

$$r_{1,3} = \frac{1}{2}(e_1 + e_5).$$

Let us consider some sparse inputs $x \in \{0,1\}^8$ with sparsity $s = 2$:
- For $x = e_1 + e_2$, we have $\mathcal{M}(x) = 0$ and thus $\mathcal{M}(x) \cdot r_{1,3} = 0 = x_1 \wedge x_3$, which is correct.
- For $x = e_1 + e_3$, we have $\mathcal{M}(x) = (1,0,0,0,1)$ and thus $\mathcal{M}(x) \cdot r_{1,3} = 1 = x_1 \wedge x_3$, which is correct.
- For $x = e_3 + e_6$, we get some interference: $\mathcal{M}(x) = (0,0,0,0,1)$ and thus $\mathcal{M}(x) \cdot r_{1,3} = \frac{1}{2}$, which has an error of $\frac{1}{2}$.

For higher sparsity, the interference effects become more pronounced, leading to higher errors. However, when the sparsity is low, with sufficient width and a calibrated probability, one can guarantee low error outside of negligible probability, which is demonstrated in the next statements.

Note 4.25. (Notation conventions):
- We usually use i, j for neuron indices.
- We usually use k, l for input feature indices.

Proposition 4.26. (Error of UAnd with basis-aligned inputs):
Let a sparsity parameter $s \in \mathbb{N}$ be fixed. Let the width $d = d(m)$ be an increasing parameter, and let $p = p(m)$ be a decreasing probability parameter such that
- $|\Gamma_2| \in \Theta(g_2(m))$ o.n.p. and
- $|\Gamma_3| \in \mathcal{O}(g_3(m))$ o.n.p.

for some sequences $g_2(m), g_3(m) > 0$ and for neuron sets $\Gamma_2 = \Gamma_2(k_1, k_2), \Gamma_3 = \Gamma_3(k_1, k_2, k_3)$ for any distinct $\{k_1, k_2, k_3\} \subseteq 1{:}m$.

Then, the Bernoulli 2-ary UAnd network from Definition 4.23 ε-linearly represents $\mathfrak{C}_{\text{UAnd}}$ on s-sparse inputs o.n.p. with the error bounded by $\varepsilon \in \mathcal{O}\left(s^2 \frac{g_3(m)}{g_2(m)}\right)$ outside of negligible probability.

Proof.
Let $\mathcal{M} : \mathbb{R}^m \to \mathbb{R}^d$ be the Bernoulli UAnd network from Definition 4.23 with weight matrix $\mathbf{W}_{\texttt{Ber}} \in \{0,1\}^{d \times m}$ and bias $b = -1$. Without loss of generality, we consider the error for the first And $x_1 \wedge x_2$. Let $r = r^{1,2} = \frac{1}{|\Gamma_2(1,2)|} 1_{\Gamma_2(1,2)}$ be the read-off vector for the And of inputs 1 and 2 from Definition 4.23.

In order for the UAnd to be ε-linearly represented in the output, we need to have $\left|\mathcal{M}(x) \cdot r_{1,2} - x_1 \wedge x_2\right| < \varepsilon$; we note that $\mathcal{M}(x) \cdot r_{1,2} \geq x_1 x_2$, so we only consider the difference $\mathcal{M}(x) \cdot r^{1,2} - x_1 x_2$.

We have two factors that determine the concentration around the expectation: The maximum total interference and the averaging effect of the read-off.

Each interference term contributes an error of at most s, as there are at most s other active inputs that can contribute in each neuron. The number of such interference terms is at most

$$\sum_{k_1 \neq 1,2, x_{k_1}=1} |\Gamma_3(1,2,k_1)| \in \mathcal{O}(s g_3(m)) \qquad \text{o.n.p.}$$

Thus, the total error here is bounded by the product, i.e.,

$$r^{1,2} \cdot \mathcal{M}(x) - x_1 x_2 \leq \frac{1}{|\Gamma_2(1,2)|} \cdot s \cdot \mathcal{O}(s g_3(m)) = \frac{\mathcal{O}(s^2 g_3(m))}{\Theta(g_2(m))} \subseteq \mathcal{O}\left(s^2 \frac{g_3(m)}{g_2(m)}\right) \qquad \text{o.n.p.}$$

Because the UAnd consists of only polynomially many ($\binom{m}{2}$) ANDs, the maximum error over all ANDs is also in $\mathcal{O}\left(s^2 \frac{g_3(m)}{g_2(m)}\right)$ o.n.p.

□

Remark 4.27.
We omitted a case distinction in the proof, which is subsumed by the bound for the other case. This also leads to a slightly improved bound in the next corollary compared to [2]. We provide an analysis of the case in Note 1.4.

The proof of Proposition 4.26 requires o.n.p. bounds on the sizes of certain neuron sets $\Gamma_K(k_1, ..., k_K)$. In order to establish these bounds, we first give a lemma that provides o.n.p. bounds for binomial random variables with sufficiently large mean.

Lemma 4.28. (O.n.p. bounds for binomial random variables):
Let $X \sim \texttt{Bin}(d, p)$ be binomially distributed with probability $p(m)$, such that the mean $\mu(m) = d(m)p(m)$ grows at least as $\mu \in \Omega(\log(m)^c)$ with $c > 1$. Then,

$$X \in \left[\mu \pm \mu^{\frac{1}{2}+\frac{1}{2c}+\nu}\right] \subseteq \Theta(\mu)$$

outside of negligible probability for any sufficiently small $\nu > 0$.

Proof.
In the following, we choose the absolute deviation as $\Delta\mathrm{x} := \mu^{c_2}$, where we define $c_2 := \frac{1}{2} + \frac{1}{2c} + \nu$. We choose ν sufficiently small such that still $c_2 < 1$ (which requires $c > 1$). We apply the Chernoff bound with $\mu = dp$ and $\delta := \frac{\Delta\mathrm{x}}{\mu} = \mu^{c_2-1}$.

Let m be sufficiently large, such that the relative deviation is $\delta = \mu^{c_2-1} < 1$, which is required for the Chernoff bound.

We thus have for all sufficiently large m that

$$\begin{aligned}
\mathbb{P}(|X - \mu| \geq \delta\mu = \Delta\mathrm{x}) &\leq 2\exp\left(-\frac{1}{3}\mu\delta^2\right) \\
&= 2\exp\left(-\frac{1}{3}\mu^{2c_2-2+1}\right) \\
&= 2\exp\left(-\frac{1}{3}\mu^{\frac{1}{c}+2\nu}\right) \\
&\in \mathcal{O}\left(\exp\left(-\frac{1}{3}\log(m)^{\overbrace{c(\frac{1}{c}+2\nu)}^{=1+2\nu c>1}}\right)\right),
\end{aligned}$$

which is negligible, as is any function of the form $\exp\left(-\log(m)^C\right)$ for $C > 1$.

□

We can now give a concrete choice of parameters to achieve a target error. The following theorem is an analogue of [2, Thm. 1].

Corollary 4.29. (Error of UAnd with basis-aligned inputs):
Let a constant sparsity parameter $s \in \mathbb{N}$ be given. Let a target error $\varepsilon(m) \in \mathcal{O}(1)$ be given. Let $d = d(m)$ be a width parameter, which grows at least as

$$d \in \Omega(\log(m)^5 \varepsilon^{-2} s^8).$$

Then, the error of the universal And circuit $\mathfrak{C}_{\mathrm{UAnd}}$ on s-sparse inputs is bounded by ε outside of negligible probability.

Proof.
We choose the probability parameter as $p := d^{-\frac{1}{4}}\varepsilon^{\frac{1}{2}}$.

By Lemma 4.28, both Γ_2 and Γ_3 grow in the order of their expectations o.n.p., i.e.,

$$|\Gamma_2(k_1, k_2)| \in \Theta(dp^2) = \Theta\left(d^{\frac{1}{2}}\varepsilon\right)$$
$$|\Gamma_3(k_1, k_2, k_3)| \in \Theta(dp^3) = \Theta\left(d^{\frac{1}{4}}\varepsilon^{\frac{3}{2}}\right)$$

o.n.p. for any indices $\{k_1, k_2, k_3\} \subseteq 1{:}m$. Note that both expectations grow sufficiently fast, as required by Lemma 4.28:

(i) For Γ_2, we have $\mathbb{E}[|\Gamma_2|] = d^{\frac{1}{2}}\varepsilon \in \Omega\big(\log(m)^{\frac{5}{2}}\big)$.
(ii) For Γ_3, we have $\mathbb{E}[|\Gamma_3|] = d^{\frac{1}{4}}\varepsilon^{\frac{3}{2}} \in \Omega\big(\log(m)^{\frac{5}{4}}\big)$.

Thus, we can apply Proposition 4.26 with $g_2(m) := dp^2$ and $g_3(m) := dp^3$ to obtain an error bound of

$$\varepsilon' \in \mathcal{O}\left(s^2\frac{dp^3}{dp^2}\right) = \mathcal{O}(s^2 p) = \mathcal{O}\big(s^2 d^{-\frac{1}{4}}\varepsilon^{\frac{1}{2}}\big) = \mathcal{O}\big(\log(m)^{-\frac{5}{4}}\varepsilon\big) \qquad \text{o.n.p.}$$

This becomes $< \varepsilon$ for sufficiently large m due to the $\log(m)^{-\frac{5}{4}}$ factor.

□

It follows naturally that there exists a network with the desired properties for sufficiently large m.

Corollary 4.30. (Existence of a network simulating UAnd with basis-aligned inputs):
For a fixed sparsity $s \in \mathbb{N}$ and sufficiently large input length m, there exists a single-layer neural network $\mathcal{M}(x) = \mathrm{ReLU}(\mathbf{W}_{\mathtt{Ber}}x + b)$ that ε-linearly represents the universal And circuit $\mathfrak{C}_{\mathrm{UAnd}}$ on at most s-sparse inputs with width $d = d_m \in \Omega(\log(m)^5\varepsilon^{-2}s^8)$ o.n.p.

Proof.
By Corollary 4.29, the maximum error over any $s' \leq s$-sparse input is in $\mathcal{O}(\varepsilon)$ outside of negligible probability. As we have only polynomially many choices for s', we can ensure that the maximum error is $< \varepsilon$ o.n.p. Thus, by choosing sufficiently large $d = d_m \in \Omega(\log(m)^5\varepsilon^{-2}s^8)$, we can ensure that the maximum error is $\varepsilon_{\mathrm{out}} < \varepsilon$ o.n.p. As the probability for such a network to exist goes to 1, there exists such a network for any sufficiently large m.

□

So far, we have assumed that the inputs are basis-aligned. To compute on superposed inputs, we extend the previous construction. For this, we need two components:

(i) **Feature encoding**: We show that there exists a feature encoding that encodes sparse inputs with low interference. This is given by the classical Johnson-Lindenstrauss lemma.
(ii) **UAnd with noised inputs**: We show that the UAnd tolerates input noise to some extent.

Lemma 4.31. (Feature encodings for sparse inputs):
Let $s \in \mathbb{N}$ be a fixed sparsity limit and $\varepsilon < 1$ a fixed interference parameter. Then, there exists a linear feature encoding $\Phi : \{0,1\}^m \to \mathbb{R}^d, x \mapsto \mathbf{\Phi}\ x$ such that $d \in \Theta(\log(m)\varepsilon^{-2}s^2)$ with maximum total interference bounded by ε, i.e.,

$$\left\|\mathbf{\Phi}^T\ \mathbf{\Phi}\ x - x\right\|_\infty < \varepsilon \qquad \forall x \in [0,1]^m \text{ with } \|x\|_0 \leq s.$$

Proof.
The Johnson-Lindenstrauss lemma for scalar products Corollary 2.9 gives us a set of m almost-unit vectors $\phi_1, ..., \phi_m \in \mathbb{R}^k$ with $k \in \Theta(\log(m)\varepsilon^{-2}s^2)$ such that the scalar products of any two vectors deviate by at most $\varepsilon' := \frac{\varepsilon-\nu}{s} > 0$ for some sufficiently small $\nu > 0$. Construct the feature encoding matrix as $\mathbf{\Phi} := (\phi_1, ..., \phi_m) \in \mathbb{R}^{k\times m}$. Then, for any s-sparse input $x \in [0,1]^m$, we have for each entry of the interference vector

$$\begin{aligned}\left|\left(\mathbf{\Phi}^T\ \mathbf{\Phi}\ x - x\right)_i\right| &= \left|\phi_i \cdot \left(\sum_{j:x_j\neq 0} x_j\phi_j\right) - x_i\right| \\ &\leq x_i\ |\phi_i\cdot\phi_i - 1| + \sum_{j:j\neq i, x_j\neq 0} x_j\ |\phi_i\cdot\phi_j| \\ &\leq s\cdot\varepsilon' = \varepsilon - \nu < \varepsilon.\end{aligned}$$

□

Note 4.32. (Symmetrizing the Noise Generated by the Feature Encoding):
The feature encoding from Lemma 4.31 is deterministic. However, it is sometimes convenient to assume the natural condition that the noise introduced by the feature encoding is random and symmetrically distributed around zero in order to show that the noise concentrates. This can be achieved by randomly flipping the signs of the feature encoding columns. More concretely, we can multiply by a random diagonal matrix $D \in \{0,+1,-1\}^{m\times m}$ with diagonal entries independently drawn from a Rademacher distribution. I.e.,

$$D_{i,i} \sim \texttt{Rad} \qquad \forall i \in 1{:}m$$

and $D_{i,j} = 0$ for all $i \neq j$. Then, we can define the symmetrized random feature encoding as $\mathbf{\Phi}_{\text{sym}} := \mathbf{\Phi}\ D$. This ensures that every interference between different features is symmetric, i.e.,

$$\mathbb{E}\left[\mathbf{\Phi}_{\text{sym}}^T\mathbf{\Phi}_{\text{sym}}\right]_{i,j} = \mathbb{E}\left[\phi_i\cdot\phi_j D_{i,i}D_{j,j}\right] = 0 \qquad \forall i,j \in 1{:}m \text{ with } \ i\neq j.$$

We now derive error bounds for inputs that include noise. This shows that the UAnd construction is robust to input noise to some extent.

Note 4.33. (Contribution and relation to [2]):
In the following, we give a statements with detailed proofs for the error of a Bernoulli UAnd layer operating on superposed inputs. The statement in Corollary 4.35 is an analogue to [2, Thm. 2], but we obtain a different asymptotic scaling of only $d \in \tilde{\mathcal{O}}\left(m^{\frac{2}{3}}\right)$ instead of $d \in \tilde{\mathcal{O}}\left(m^{\frac{1}{2}}\right)$. The original proof sketches in [2, Thm. 2] only gives the construction, chaining the feature encoding with the UAnd layer, but does not give error computations. It states that "Carefully tracking error terms shows that we need $d = \tilde{\Theta}(\sqrt{m})$ neurons," but does not provide further details. The analogous proofs in their appendix, in [2, Thm. 11, Thm. 14], use different constructions and significantly different proof approaches.

Notably, there are additional issues, which arise when using the Bernoulli construction here: Due to the polynomial width scaling, we need to use a different o.n.p. bound for binomial random variables with decreasing expectation (see Lemma 4.36). Because of this, we have split the theorem into several parts, giving a more general formulation in Proposition 4.34 that allows to choose p and d more freely. This allows us to choose p and d differently for different parameter regimes and not include many special cases in the theorem statement.

Proposition 4.34. (Error propagation of the Bernoulli UAnd layer):
Let one-layer Bernoulli UAnd network $\mathcal{M}(x) = \mathrm{ReLU}(\mathbf{W}_{\text{in}}x + b)$ from Definition 4.23 with weight matrix $\mathbf{W}_{\text{in}} \in \{0,1\}^{d\times m}$ and bias $b := -1$ be given, such that it represents the universal And circuit $\mathfrak{C}_{\text{UAnd}}$ on s-sparse inputs with error bounded by ε_0 outside of negligible probability. Let $\tilde{x} = x + \Delta\mathrm{X}$ be a noised input with $\Delta\mathrm{X}$ being random noise. Let this noise satisfy the growth condition

$$\sum_{i=1}^{m} \Delta\mathrm{X}_i B_i \in \mathcal{O}(g_{\Delta\mathrm{X}}(m)) \qquad \text{o.n.p.}$$

for $B_i \sim \texttt{Ber}(p)$ i.i.d. for $i \in 1{:}m$ and some function $g_{\Delta\mathrm{X}}(m) > 0$.

Then, the output error for the noised input is bounded by $\varepsilon_{\text{out}} \in \mathcal{O}(\varepsilon_0 + g_{\Delta\mathrm{X}}(m))$ o.n.p.

Proof.
Consider the output of one neuron. Because the weights are Bernoulli-distributed, each neuron has $\sum_i B_i$ connections for some $B_i \sim \texttt{Ber}(p)$. Each connection contributes noise of at most $\Delta\mathrm{X}_i$. Thus, the total noise contribution per neuron is distributed as $\sum_{i=1}^{m} \Delta\mathrm{X}_i B_i \in \mathcal{O}(g_{\Delta\mathrm{X}}(m))$ o.n.p. by assumption. As we read off from $|\Gamma_2|$ neurons and average over them, the total contribution to the error is also $\mathcal{O}(g_{\Delta\mathrm{X}}(m))$ o.n.p.

□

This allows us to chain the feature encoding with the UAnd computation to obtain a layer which operates directly on superposed inputs, giving us an analogous result to [2, Thm. 2].

Corollary 4.35. (UAnd with inputs in superposition):
Let $s \in \mathbb{N}$ be a fixed sparsity limit and $\varepsilon < 1$ a fixed interference parameter, which is allowed to decrease logarithmically, i.e., $\varepsilon \in \Theta(\log(m)^{-c_\varepsilon})$ for $c_\varepsilon > 0$. Then there exists a feature encoding Φ and single-layer neural net $\mathcal{M}(x) = \mathrm{ReLU}(\mathbf{W}_{\text{in}}x + b)$ with input size and width $d = \tilde{\mathcal{O}}\left(s^{\frac{4}{3}} m^{\frac{2}{3}} \varepsilon^{-\frac{5}{3}}\right)$, where $\mathcal{M} \circ \Phi$ ε-linearly represents the universal And circuit $\mathfrak{C}_{\text{UAnd}}$ on all s-sparse inputs x.

To prove Corollary 4.35, we need to establish o.n.p. bounds for binomial random variables with decreasing expectation. Note that Chernoff bounds do not yield o.n.p. bounds in this case, as this requires the expectation to grow. However, we can still obtain o.n.p. bounds via a union bound argument.

Lemma 4.36. (O.n.p. upper bounds for binomial random variables):
Let $X = \sum_i X_i$ with $X_i \sim \texttt{Ber}(p)$ being i.i.d. Bernoulli random variables with probability $p = p(n)$ and expectation $\mu = \mu(n) = np(n)$.

- If $\mu < 1$ for all sufficiently large n, we have $X \in \mathcal{O}(\log(n)^{1+\varepsilon})$ o.n.p.
- If $\mu \in \mathcal{O}(n^{-\alpha})$, we have $X \in \mathcal{O}(\log(n)^{\varepsilon})$ for any $\varepsilon > 0$ o.n.p.

for any $\varepsilon > 0$.

Proof.
We have for any $k \in \mathbb{N}$ that

$$\mathbb{P}(X \geq k) \leq \binom{n}{k} p^k \in \mathcal{O}\left(n^k p^k\right) = \mathcal{O}\left((np)^k\right) = \mathcal{O}(\exp(k \log(\mu))).$$

- If $\mu < 1$, then $\log(\mu) < 0$. Thus, $\exp(\log(n)^{1+\varepsilon}\log(\mu))$ is negligible (similarly to Example 4.2).
- If $\mu \in \mathcal{O}(n^{-\alpha})$, then $\log(\mu) \leq -\alpha\log(n)$ for some $\alpha > 0$ and sufficiently large n. Thus, $\exp(\log(n)^{\varepsilon}\log(\mu)) \leq \exp(-\alpha\log(n)^{1+\varepsilon})$ is again negligible for any $\varepsilon > 0$.

□

Proof of Corollary 4.35.
In order to construct a network that operates on superposed inputs, we chain the feature encoding from Lemma 4.31 with the basis-aligned Bernoulli UAnd layer from Definition 4.23. I.e. we construct the weight matrix $\mathbf{W}_{\text{in}}$ as product of the Bernoulli matrix $\mathbf{W}_{\texttt{Ber}}$ from Definition 4.23 and the symmetrized feature encoding matrix $\mathbf{\Phi}_{\text{sym}}$ from Note 4.32. I.e.,

$$\mathbf{W}_{\text{in}} := \underbrace{\mathbf{W}_{\texttt{Ber}}}_{\in \mathbb{R}^{d\times m}} \mathbf{\Phi}_{\text{sym}}^{T} \in \mathbb{R}^{d\times d}.$$

Here we choose the Bernoulli probability parameter as $p := \log(m)^2 d^{-\frac{1}{2}}\varepsilon^{-\frac{1}{2}} \in \tilde{\Theta}\left(d^{-\frac{1}{2}}\varepsilon^{-\frac{1}{2}}\right)$.

Let ε_{in} be the interference bound of the feature encoding. From Corollary 2.9, we have that the incoming interference error is $\varepsilon_{\text{in}} \in \mathcal{O}\left(s\log(m)d^{-\frac{1}{2}}\right) \subseteq \tilde{\mathcal{O}}\left(sd^{-\frac{1}{2}}\right)$.

We now analyze the concrete error bounds of this construction by applying Proposition 4.34. For this, we first need to bound the effect of the noise generated by the feature encoding. Let $B_k \sim \texttt{Ber}(p)$ and $B := \sum_k^m B_k \Delta \mathrm{X}_k$. First, note that by Chernoff bounds, we have $\sum_k B_k \in \Theta(mp)$ o.n.p. Thus, we can apply Hoeffding's inequality on the independent random variables $\{B_k\Delta \mathrm{X}_k\}_k$ to obtain

$$\begin{aligned}\mathbb{P}\left(\left|\sum_k \Delta \mathrm{X}_k B_k\right| > \Delta \mathrm{x}\right) &\leq 2\exp\left(-2\frac{\Delta \mathrm{x}^2}{\sum_k B_k \varepsilon_{\text{in}}^2}\right)\\ &\leq 2\exp\left(-2\frac{\Delta \mathrm{x}^2}{\Theta(mp)\varepsilon_{\text{in}}^2}\right) \qquad \text{o.n.p.}\end{aligned}$$

Thus, it follows that the sum B is o.n.p. bounded by

$$B \in \mathcal{O}\left(\varepsilon_{\text{in}}(mp)^{\frac{1}{2}}\log(m)\right) \subseteq \tilde{\mathcal{O}}\left(sd^{-\frac{1}{2}}m^{\frac{1}{2}}d^{-\frac{1}{4}}\varepsilon^{-\frac{1}{4}}\right) = \tilde{\mathcal{O}}\left(sm^{\frac{1}{2}}d^{-\frac{3}{4}}\varepsilon^{-\frac{1}{4}}\right).$$

To apply Proposition 4.34, we also need to bound the sizes of the neuron sets Γ_2 and Γ_3. For Γ_2, we have

$$\mathbb{E}[|\Gamma_2|] = dp^2 = \log(m)^4\varepsilon^{-1} \in \Omega(\log(m)^4).$$

Thus, the expectation grows sufficiently fast to apply Lemma 4.28, giving us that $|\Gamma_2| \in \Theta(dp^2)$ o.n.p..

As d is polynomial in m, we have that

$$\mathbb{E}[dp^3] = \log(m)^6\varepsilon^{-\frac{3}{2}}d^{-\frac{1}{2}} \in \tilde{\mathcal{O}}\left(d^{-\frac{1}{2}}\right))$$

is polynomially decreasing in m. Thus, by Lemma 4.36, we have that $|\Gamma_3| \in \mathcal{O}(\log(m)^{\nu})$ o.n.p. for any $\nu > 0$.

Now, Proposition 4.34 gives us that the output error is bounded by the error of the basis-aligned UAnd layer plus the noise contribution from the feature encoding. Thus, by choosing $d \in \tilde{\Theta}\left(m^{\frac{2}{3}}\varepsilon^{-\frac{5}{3}}s^{\frac{4}{3}}\right))$ with sufficiently large polylogarithmic factor, we have:

- The interference noise contribution is bounded by $g_{\Delta X} \in \tilde{\mathcal{O}}\left(m^{\frac{1}{2}} d^{-\frac{3}{4}} \varepsilon^{-\frac{1}{4}}\right) \subseteq \mathcal{O}(\log(m)^{-c}\varepsilon)$ for some $c > 0$. Thus, it holds $g_{\Delta X} < \frac{\varepsilon}{2}$ for sufficiently large m.
- The basis-aligned UAnd error is bounded by $\varepsilon_0 \in \tilde{\mathcal{O}}\left(s^2 \frac{\log(m)^\nu}{p^2 d}\right) \subseteq \mathcal{O}(\log(m)^{-c}\varepsilon)$ for some $c > 0$. Thus it holds $\varepsilon_0 < \frac{\varepsilon}{2}$ for sufficiently large m.

Thus, the total output error is bounded as $\varepsilon_{\text{out}} < \varepsilon$ o.n.p. for sufficiently large m.

□

Remark 4.37. (Computational vs. representational bounds on superposition):
Note that for this theorem, we require the (both input and output) width to scale as $m^{\frac{2}{3}}$ in the number of features, instead of polylogarithmically as in Corollary 4.30.

Note that this 'computational bound with noise' for the width is different from the both the 'pure computational bound' (Corollary 4.29), where we do not have incoming interference errors, and the 'representational bound' for the width required to only represent s-sparse vectors with low interference, which in principle allows logarithmic scaling of the width d in the number of features m.

Notably, the polynomial sublinear scaling bound appears to be significantly more realistic, as we expect read-off error and other sources of error in practice.

4.2.2 UAnd in Gaussian-initialized MLPs

In this section, we show that superposed UAnd circuit is surprisingly universally computable—it can be read off from a sufficiently large Gaussian-initialized layer output. While the previous constructions were based on more artificially sampled random matrices, which are quite different from standard initializations, Gaussian initializations are widely used in practice. As the argument does only depend on concentration, we expect similar results to hold for other i.i.d. random initialization schemes.

Remark 4.38. (Contribution and differences):
We have significantly extended the original theorem proof sketch to a full proof. The authors provide a proof of a similar, but different statement in their appendix. Their statement in the appendix assumes a polynomial-in-input scaling of the width, which is not sufficient for the actual theorem in their main text. This leads to the weak Bernstein bound being insufficient. We thus provide a different proof here. This proof invokes Gaussian-Lipschitz concentration with $d^{\frac{1}{2}+\text{const}}$ bound, which allows us to prove the theorem with logarithmic width scaling.

Theorem 4.39. (Randomly Initialized MLPs Linearly Represent UAnd, [2, Thm. 3]):
Let $\mathcal{M} : \mathbb{R}^m \to \mathbb{R}^d$ be a one-layer MLP with $d = \tilde{\Omega}_m\left(\frac{1}{\varepsilon^2}\right)$ neurons that takes input x, with weights $(W_{\text{in}})_{i,j}$ drawn i.i.d from a normal distribution $\mathcal{N}(0, \delta^2)$ and $b = 0$.

This MLP ε-linearly represents the universal And circuit $\mathfrak{C}_{\text{UAnd}}$ on s-sparse inputs outside of negligible probability.

We structure the proof of the theorem into two main lemmata:
(i) We show that the expected read-off equals the And of inputs in Lemma 4.40.
(ii) We show that the deviation of the read-off from its expectation is bounded outside of negligible probability in Lemma 4.42.

Together, these imply the theorem.

Lemma 4.40. (Gaussian initialized MLPs: UAnd expectation):
Without loss of generality, consider $\delta = 1$. With the assumptions from Theorem 4.39, consider an And $x_k \wedge x_l$ indexed by $1 \leq k < l \leq m$. We define the unnormalized read-off vector $\tilde{r}^{k,l}$ first. We count the contribution of a row i positive if the corresponding two weights W_{ik}, W_{il} have the same sign and negative otherwise. I.e.,

$$\tilde{r}_i^{k,l} := \operatorname{sgn}(W_{ik}W_{il}) \qquad \forall i \in 1{:}d.$$

Then, there exists a well-defined normalization factor $\eta\big(w^k, w^l\big) > 0$, such that the expected read-off matches the And of inputs. I.e.,

$$\mathbb{E}[\mathcal{M}(x) \cdot r] = x_k \wedge x_l$$

for all k, l with $1 \leq k < l \leq d$ and $x \in \{0,1\}^m$, where the read-off vector is $r^{k,l} := \eta^{-1}\tilde{r}^{k,l}$ and $\eta := \mathbb{E}[\tilde{r} \cdot \mathcal{M}(x)]$ for an x with $x_k = x_l = 1$ and $\|x\|_0 = s$.

In order to prove the theorem, we record a simple representation of the And. We want to express the read-off in a form that separates the random signs from the absolute values of the weights. In both parts of the proof, Lemma 4.40 and Lemma 4.42, separating the signs and absolute values is fundamental.

Lemma 4.41. (Representation of read-off):
With the assumptions from Theorem 4.39 and Lemma 4.40, consider w.l.o.g. the first And $x_1 \wedge x_2$. Then, the unnormalized read-off has the form

$$\mathcal{M}(x) \cdot \tilde{r}^{1,2} = \sum_i^d R_{i1}R_{i2}\,\mathrm{ReLU}\left(R_{i1}|W_{i1}|x_1 + R_{i2}|W_{i2}|x_2 + \sum_{j\geq 3} W_{ij}x_j\right),$$

where $R_{ik} := \operatorname{sgn}(W_{ik})$ for $k \in \{1,2\}$ are Rademacher random variables a.s.

Proof of Lemma 4.41.
This follows directly by expanding the definition of $\tilde{r}$ and defining $R_{ik} := \operatorname{sgn}(W_{ik})$ for $k \in \{1,2\}$.

□

Proof of Lemma 4.40.
We again consider w.l.o.g. the first And $x_1 \wedge x_2$ with corresponding read-off $r := r^{1,2}$. Let $x \in \{0,1\}^m$ be a fixed s-sparse input vector. We are interested in the expectation of the unnormalized read-off from Lemma 4.41. Because the weight distributions are equal, we can reduce this further to

$$\begin{aligned}\mathbb{E}[\mathcal{M}(x) \cdot \tilde{r}] &= \mathbb{E}\left[\sum_i^d R_{i1}R_{i2}\,\mathrm{ReLU}\left(R_{i1}|W_{i1}|x_1 + R_{i2}|W_{i2}|x_2 + \underbrace{\sum_{j\geq 3} W_{ij}x_j}_{\sim\mathcal{N}(0,\sigma^2\in\Theta(s))}\right)\right] \\ &= d\;\mathbb{E}[R_{11}R_{12}\,\mathrm{ReLU}(R_{11}|W_{11}|x_1 + R_{12}|W_{12}|x_2 + Z)],\end{aligned}$$

where $Z \sim \mathcal{N}(0, \sigma^2 \in \Theta(s))$.

For readability, we now denote $R_1 := R_{11}, R_2 := R_{12}, X := |W_{11}|, Y := |W_{12}|$.

Now, we have the three cases for the And of x_1, x_2:
- If both $x_1 = x_2 = 0$, then $\mathbb{E}[\mathcal{M}(x) \cdot \tilde{r}] = d\;\mathbb{E}[R_1R_2\,\mathrm{ReLU}(Z)] = 0 = x_1 \wedge x_2$.

- If $x_1 + x_2 = 1$, let w.l.o.g. $x_1 = 1$. Then,

$$\begin{aligned}\mathbb{E}[\mathcal{M}(x) \cdot \tilde{r}] &= d\ \mathbb{E}[R_1 R_2\, \mathrm{ReLU}(R_1 X + Z)] \\ &= d\underbrace{\mathbb{E}[R_2]}_{=0}\ \mathbb{E}[R_1\, \mathrm{ReLU}(R_1 X + Z)] = 0.\end{aligned}$$

- If both $x_1 = x_2 = 1$, showing that the expectation is positive is non-trivial.

We now consider the last case in detail. For readability, we are going to use the short-hand notation $(\cdot)_+ = \mathrm{ReLU}(\cdot)$.

Let us denote the expectation term as $E_{\mathbb{E}} \coloneqq \mathbb{E}\Big[R_1 R_2 (R_1 X + R_2 Y + Z)_+\Big]$.

We want to show that $E_{\mathbb{E}} > 0$.

Because the expectation is symmetric in X, Y, we first reduce it as

$$E_{\mathbb{E}} = 2\ \mathbb{E}\Big[R_1 R_2 (R_1 X + R_2 Y + Z)_+ 1_{X \le Y}\Big].$$

to reduce the number of case distinctions.

We evaluate the expectation partially over R_1, R_2 which allows to see cancellations more clearly. This yields

$$\frac{1}{2} E_{\mathbb{E}} = \mathbb{E}\left[1_{X \le Y} \underbrace{\big((+X + Y + Z)_+ - (-X + Y + Z)_+ - (+X - Y + Z)_+ + (-X - Y + Z)_+\big)}_{=:E}\right].$$

We now want use a case distinction of Z to eliminate the ReLUs. The larger Z is compared to X, Y, the more ReLUs are active. We split into five cases:

$$\begin{aligned}\frac{1}{2} E_{\mathbb{E}} &= \mathbb{E}[1_{X \le Y}(1_{Z \le -X-Y} + 1_{-X-Y < Z \le X-Y} + 1_{X-Y < Z \le -X+Y} + 1_{-X+Y < Z \le X+Y} + 1_{Z > X+Y}) E] \\ &=: E_{\mathbb{E}}^{--} + E_{\mathbb{E}}^{-} + E_{\mathbb{E}}^{0} + E_{\mathbb{E}}^{+} + E_{\mathbb{E}}^{++},\end{aligned}$$

where $E_{\mathbb{E}}^{--}$ corresponds to $\mathbb{E}[1_{X \le Y} 1_{Z \le -X-Y} E]$ and so on.

Notably, each of these expectations is non-negative:

(i) $E_{\mathbb{E}}^{--} = 0$ because all ReLUs are inactive.

(ii) $E_{\mathbb{E}}^{-}$ is non-negative because only the first ReLU is active, which contributes positively.

(iii) For $E_{\mathbb{E}}^{0}$ we have

$$E_{\mathbb{E}}^{0} = \mathbb{E}\left[1_{X \le Y} 1_{X-Y < Z \le -X+Y} \underbrace{\big((+X + Y + Z) - (-X + Y + Z)\big)}_{=2X}\right] = \mathrm{const}(s) > 0.$$

(iv) For $E_{\mathbb{E}}^{+}$ we have

$$E_{\mathbb{E}}^{+} = \mathbb{E}\left[1_{X \le Y} 1_{-X+Y < Z \le X+Y} \underbrace{\big((+X + Y + Z) - (-X + Y + Z) - (+X - Y + Z)\big)}_{=X+Y-Z}\right] \ge 0,$$

due to the condition $Z \le X + Y$.

(v) Finally, for $E_{\mathbb{E}}^{++}$ we have

$$E_{\mathbb{E}}^{++} = 0,$$

as all ReLUs are active and the terms cancel out.

Thus, we have $E_{\mathbb{E}} > 0$. This shows that for $x_1 = x_2 = 1$, we have $\mathbb{E}[\mathcal{M}(x) \cdot \tilde{r}] = d \cdot E_{\mathbb{E}} > 0$, which defines $\eta := \mathbb{E}[\mathcal{M}(x) \cdot \tilde{r}] > 0$. By the symmetry of the distributions for different choices of x with $x_k = x_l = 1$ and $\|x\|_0 = s$, the normalization factor is well-defined.

□

Lemma 4.42. (Gaussian-initialized-MLPs: Bounded deviation):
With the assumptions from Lemma 4.40, consider the And x_k, x_l. Then, for any fixed s-sparse input $x \in \{0,1\}^m$, we have that outside of negligible probability

$$\mathcal{M}(x) \cdot r^{k,l} \in [\mathbb{E}[\mathcal{M}(x) \cdot r^{k,l}] \pm \Theta(d^C)]$$

for some constant $\frac{1}{2} < C < 1$ outside of negligible probability w.r.t. m.

Note 4.43. (Gaussian o.n.p. bounds):
The probability that any of the polynomially many W_{ik} deviates by more than d^ε for any $\varepsilon > 0$ is negligible w.r.t. m. I.e., $\|\boldsymbol{W}\|_\infty \leq d^\varepsilon$ o.n.p.

Proof.
By standard Gaussian tail bounds Note 1.1, we have for any single Gaussian variable $W_{ik} \sim \mathcal{N}(0,1)$ that

$$\begin{aligned}\mathbb{P}(|W_{ik}| \geq \Delta\mathrm{x} := d^\varepsilon) &= 2 - 2\Phi(\Delta\mathrm{x}) \leq 2\frac{\phi(\Delta\mathrm{x})}{\Delta\mathrm{x}} \\ &\in \mathcal{O}\left(\frac{\exp\left(-\frac{d^{2\varepsilon}}{2}\right)}{d^\varepsilon}\right) \subseteq \mathcal{O}\left(\exp\left(-\frac{\log(m)^{2\varepsilon}}{2}\right)\right),\end{aligned}$$

which is negligible w.r.t. m.

□

Proof of Lemma 4.42.

Without loss of generality, consider again the first And $x_1 \wedge x_2$. In order to prove the statement, we need to show that the read-off $\mathcal{M}(x) \cdot r$ is sufficiently concentrated around its expectation. However, we cannot simply apply the Gaussian concentration theorem Theorem 4.12, because the term is not Lipschitz-continuous in all Gaussian variables due to the sign dependencies.

Thus, we use a two-step approach: First, the Hoeffding inequality Theorem 4.8 to show concentration for the sign variables, and then apply the Gaussian concentration theorem to the Gaussian variables.

For this, it is convenient to express the read-off in terms of sign variables and Gaussians, where the sign variables are fully independent of the Gaussian values. I.e., we represent $\boldsymbol{W} = \left(R^1 \odot \left|\widetilde{W}^1\right|, R^2 \odot \left|\widetilde{W}^2\right|, \widetilde{W}^3, ..., \widetilde{W}^m\right)$, where $\widetilde{W}$ is a matrix of i.i.d. Gaussian variables, which are independent of the Rademacher sign matrix $R \in \{-1,1\}^{d \times 2}$.

We denote the read-off random variable as Z_d. As of Lemma 4.41 and the $\widetilde{W}$ definition, we have

$$Z_d = \sum_i^d R_{i1} R_{i2} \operatorname{ReLU}\left(R_{i1}\left|\widetilde{W}_{i1}\right| x_1 + R_{i2}\left|\widetilde{W}_{i2}\right| x_2 + \sum_{j=3, x_j=1}^m \widetilde{W}_{ij}\right).$$

Hoeffding in the form of Note 4.9 now gives us a bound on the deviation of Z_d from $\mathbb{E}_R[Z_d]$ for fixed $\widetilde{W}$, given an o.n.p. bound on $\left\|\widetilde{\boldsymbol{W}}\right\|_\infty$. Let $\frac{1}{4} > \nu > 0$ be a small constant. By Note 4.43, we have $\left\|\widetilde{\boldsymbol{W}}\right\|_\infty \leq d^\nu$ o.n.p. Thus, each individual summand $|Z_d - Z_{d-1}|$ is bounded by $c = 2\,\mathrm{ReLU}(sd^\nu) = 2sd^\nu$. We choose the deviation to be $\Delta\mathrm{x} := d^C$ for the constant $C := \frac{1}{2} + 2\nu$. Then, the Hoeffding inequality yields

$$\begin{aligned}\mathbb{P}\Big(|Z_d - \mathbb{E}_R[Z_d]| \geq \Delta\mathrm{x} \mid \widetilde{\boldsymbol{W}}, \left\|\widetilde{\boldsymbol{W}}\right\|_\infty \leq d^\nu\Big) &\leq 2\exp\left(-\frac{1}{2}\frac{\Delta\mathrm{x}^2}{\sum_i^d c^2}\right)\\ &= 2\exp\left(-\frac{1}{2}\frac{d^{2C}}{dc^2}\right)\\ &= 2\exp\left(-\frac{1}{8s^2}d^{2(C-\nu)-1}\right) = 2\exp\left(-\frac{1}{8s^2}d^{2\nu}\right),\end{aligned}$$

which is negligible w.r.t. m.

We now need to bound the deviation of $\mathbb{E}_R[Z_d]$ from $\mathbb{E}[Z_d]$. For this, we use the Gaussian concentration theorem Theorem 4.12. The partial expectation $\mathbb{E}_R[Z_d]$ gives us a function

$$\begin{aligned}\mathbb{E}_R[Z_d] &= \sum_i^d \frac{1}{4}\sum_{r_1,r_2\in\{-1,1\}} r_1 r_2\,\mathrm{ReLU}\left(r_1\left|\widetilde{W}_{i1}\right|x_1 + r_2\left|\widetilde{W}_{i2}\right|x_2 + \sum_{k=3,x_k=1}^m \widetilde{W}_{ik}\right)\\ &=: f\left(\widetilde{\boldsymbol{W}}\right).\end{aligned}$$

This function is Lipschitz-continuous: Consider two matrices $\widetilde{\boldsymbol{W}}, \widetilde{\boldsymbol{W}}' \in \mathbb{R}^{d\times m}$. Then, we have

$$\begin{aligned}&\left|f\left(\widetilde{\boldsymbol{W}}'\right) - f\left(\widetilde{\boldsymbol{W}}\right)\right| && \downarrow \text{expansion, 1-Lipschitz continuity of ReLU}\\ &\leq \sum_i^d \frac{1}{4}\sum_{r_1,r_2\in\{-1,1\}}\left|\left(r_1\left(\left|\widetilde{W}'_{i1}\right| - \left|\widetilde{W}_{i1}\right|\right)x_1 + r_2\left(\left|\widetilde{W}'_{i2}\right| - \left|\widetilde{W}_{i2}\right|\right)x_2 + \sum_{k=3,x_k=1}^m\left(\widetilde{W}'_{ik} - \widetilde{W}_{ik}\right)\right)\right|\\ &\leq \sum_i^d \frac{1}{4}\sum_{r_1,r_2\in\{-1,1\}}\left|\widetilde{W}'^i - \widetilde{W}^i\right|\cdot x && \downarrow \text{Sum reduction, Cauchy-Schwarz}\\ &\leq \sum_i^d \frac{1}{4}4\sqrt{s}\left\|\widetilde{W}'^i - \widetilde{W}^i\right\|_2 && \downarrow \text{Cauchy-Schwarz}\\ &\leq \sqrt{ds}\left\|\widetilde{\boldsymbol{W}}' - \widetilde{\boldsymbol{W}}\right\|_F,\end{aligned}$$

i.e., the Lipschitz constant is $L = \sqrt{ds}$. Thus, by Theorem 4.12, we obtain

$$\begin{aligned}\mathbb{P}(|\mathbb{E}_R[Z_d] - \mathbb{E}[Z_d]| \geq \Delta\mathrm{x}) &\leq 2\exp\left(-\frac{\Delta\mathrm{x}^2}{4L^2}\right) = 2\exp\left(-\frac{d^{2C}}{4ds}\right)\\ &= 2\exp\left(-\frac{d^{1+4\nu-1}}{4s}\right) \in \mathcal{O}\left(\exp\left(-\frac{d^{4\nu}}{4s}\right)\right),\end{aligned}$$

which is negligible w.r.t. m.

Finally, we can combine both bounds:

$$\begin{aligned}
&\mathbb{P}(|Z_d - \mathbb{E}[Z_d]| \geq 2\Delta\mathrm{x}) \\
\leq\ &\mathbb{P}\Big(\left\|\widetilde{\boldsymbol{W}}\right\|_\infty \geq d^\nu\Big) + \mathbb{P}\Big(|Z_d - \mathbb{E}_R[Z_d]| \geq \Delta\mathrm{x} \mid \widetilde{\boldsymbol{W}}, \left\|\widetilde{\boldsymbol{W}}\right\|_\infty \leq d^\nu\Big)\ \mathbb{P}\Big(\left\|\widetilde{\boldsymbol{W}}\right\|_\infty \leq d^\nu\Big) \\
&+ \mathbb{P}(|\mathbb{E}_R[Z_d] - \mathbb{E}[Z_d]| \geq \Delta\mathrm{x}) \\
\leq\ &\text{negligible} + \text{negligible} + \text{negligible},
\end{aligned}$$

which is negligible w.r.t. m.

□

Proof of Theorem 4.39.
By Lemma 4.40, we have that the expected read-off matches the And of inputs. By Lemma 4.42, we have that the read-off is concentrated around its expectation up to $\Theta\big(d^{C}\big)$. Thus, outside of negligible probability, we have for the first And

$$\mathcal{M}(x) \cdot r = \frac{1}{\eta}[\eta x_1 \wedge x_2 \pm \Theta\big(d^{C}\big)] \subseteq \left[x_1 \wedge x_2 \pm \Theta\left(d^{\overbrace{<0}^{C-1}}\right)\right] \text{ o.n.p.},$$

and by choosing $d = d_m \in \Theta\big(\frac{\log(m)}{\varepsilon^2}\big)$ with a sufficiently large constant factor, the error is within $[\pm\varepsilon]$ o.n.p.

□

4.2.3 UAnd of Arbitrary Arity

One can not only represent And circuits, but arbitrary Boolean circuits in superposition. As illustrated in Proposition 4.20, any Boolean circuit can be represented as a linear combination of And circuits. Thus, the first step is to extend the UAnd theorem to And circuits of arbitrary (but sufficiently small) arity.

The following is a generalization of Proposition 4.26 to ANDs of arbitrary arity.

Proposition 4.44. (Error of UAnd with basis-aligned inputs):
Let a sparsity parameter $s \in \mathbb{N}$ be fixed. Let the width $d = d(m)$ be an increasing parameter, and let $p = p(m)$ be a decreasing probability parameter such that

- $|\Gamma_n| \in \Theta(g_n(m))$ o.n.p. and
- $\left|\Gamma_{n+1}\right| \in \mathcal{O}(g_{n+1}(m))$ o.n.p.

for some sequences $g_n(m), g_{n+1}(m) > 0$ and for neuron sets $\Gamma_n = \Gamma_n(k_1, ..., k_n), \Gamma_{n+1} = \Gamma_{n+1}(k_1, ..., k_{n+1})$ for $\{k_1, ..., k_{n+1}\} \subseteq 1{:}m$ being distinct feature indices.

Then, the Bernoulli UAnd network from Definition 4.23 ε-linearly represents $\mathfrak{C}^n_{\mathrm{UAnd}}$ on s-sparse inputs o.n.p. with the error bounded by $\varepsilon \in \mathcal{O}\Big(s^2 \frac{g_{n+1}(m)}{g_n(m)}\Big)$ outside of negligible probability.

Proof.
The proof is analogous to the argument for the universal 2-AND Proposition 4.26. We consider w.l.o.g. the And of the first n inputs $x_1 \wedge x_2 \wedge \cdots \wedge x_n$.

Let $r = r^{1,\ldots,n}$ be the read-off vector. In order for the UAnd to be ε-linearly represented in the output, we need to have

$$|\mathcal{M}(x) \cdot r - x_1 \wedge x_2 \wedge \cdots \wedge x_n| < \varepsilon \qquad \text{o.n.p.}$$

Note that $\mathcal{M}(x) \cdot r \geq x_1 x_2 ... x_n$, because all interference contributions are non-negative. Thus, we only consider the difference $\mathcal{M}(x) \cdot r - x_1 x_2 ... x_n$.

The interference contributions arise from neurons that are connected to the n of the target inputs and at least one additional input x_k with $k > n$ and $x_k = 1$. There are at most

$$\sum_{x_k=1} \left|\Gamma_{n+1}(k_1, k_2, ..., k_n, k)\right| \in \mathcal{O}(s g_{n+1}(m))$$

such neurons, each contributing at most $s - (n-1) \leq s$ to the error. As we normalize by $|\Gamma_n(k_1, k_2, ..., k_n)| \in \Theta(g_n(m))$, the total interference error is thus o.n.p. bounded as

$$\frac{\mathcal{O}(s^2 g_{n+1}(m))}{\Theta(g_n(m))} = \mathcal{O}\left(s^2 \frac{g_{n+1}(m)}{g_n(m)}\right).$$

Because the UAnd consists of only polynomially many ($\binom{m}{n} \in \mathcal{O}(m^n)$) ANDs, we have the same bound for the maximum error over all ANDs o.n.p.

□

Corollary 4.45. (UAnd for arbitrary arity):
Let the arity $n \in \mathbb{N}, n \geq 2$ be constant. Let $0 < \varepsilon < 1$ be a non-increasing error parameter. Let d be a width parameter that grows at least as

$$d \in \Omega\left(\log(m)^{n+3} s^{2(n+2)} \varepsilon^{-(n+1)\frac{n+2}{n}}\right).$$

Then there exists a random construction of a single-layer neural network $\mathcal{M}(x) = \mathrm{ReLU}(\mathbf{W}_{\text{in}} x + b)$ with input size m and width d that ε-linearly represents the universal And circuit $\mathfrak{C}^n_{\text{UAnd}}$ on all s-sparse inputs x outside of negligible probability w.r.t. m.

Proof.
Analogously to Corollary 4.29, we apply Proposition 4.44 with the appropriate parameters and compute the growth bounds for the neuron sets Γ_n, Γ_{n+1}. We set the probability parameter to $p := d^{-\frac{1}{n+2}} \varepsilon^{\frac{1}{n}}$. For applying Lemma 4.28, we need to verify that the expectations $|\Gamma_n|, \left|\Gamma_{n+1}\right|$ grow sufficiently quickly. The expectations are

$$\mathbb{E}[|\Gamma_n|] = p^n d \in \Theta\left(d^{\frac{2}{n+2}} \varepsilon\right) \subseteq \Omega\left(\log(m)^{2\frac{n+3}{n+2}}\right),$$
$$\mathbb{E}\left[\left|\Gamma_{n+1}\right|\right] = p^{n+1} d \in \Theta\left(d^{\frac{1}{n+2}} \varepsilon^{\frac{n+1}{n}}\right) \subseteq \Omega\left(\log(m)^{\frac{n+3}{n+2}}\right).$$

We see that both expectations grow as $\log(m)^{1+\nu}$ for some $\nu > 0$. Thus, both $|\Gamma_n|, \left|\Gamma_{n+1}\right|$ concentrate o.n.p. around their expectations as $|\Gamma_n| \in \Theta(p^n d), \left|\Gamma_{n+1}\right| \in \Theta(p^{n+1} d)$ by Lemma 4.28.

From Proposition 4.44, we thus have that the Bernoulli UAnd network ε-linearly represents $\mathfrak{C}^n_{\text{UAnd}}$ on s-sparse inputs o.n.p. with the error ε_{out} bounded as

$$\varepsilon_{\text{out}} \in \mathcal{O}\left(s^2 \frac{p^{n+1} d}{p^n d}\right) = \mathcal{O}(s^2 p) = \mathcal{O}\left(s^2 d^{-\frac{1}{n+2}} \varepsilon^{\frac{1}{n}}\right) \qquad \text{o.n.p.}$$

To verify that this error is at most ε, we substitute the growth condition $d \in \Omega\left(\log(m)^{n+3} s^{2(n+2)} \varepsilon^{-(n+1)\frac{n+2}{n}}\right)$, which gives

$$d^{-\frac{1}{n+2}} \in \mathcal{O}\left(\left(\log(m)^{n+3} s^{2(n+2)} \varepsilon^{-(n+1)\frac{n+2}{n}}\right)^{-\frac{1}{n+2}}\right) = \mathcal{O}\left(\log(m)^{-\frac{n+3}{n+2}} s^{-2} \varepsilon^{\frac{n+1}{n}}\right).$$

Therefore, the error is bounded by

$$s^2 d^{-\frac{1}{n+2}} \varepsilon^{\frac{1}{n}} \in \mathcal{O}\left(s^2 \log(m)^{-\frac{n+3}{n+2}} s^{-2} \varepsilon^{\frac{n+1}{n}} \varepsilon^{\frac{1}{n}}\right) = \mathcal{O}\left(\log(m)^{-\frac{n+3}{n+2}} \varepsilon^{1+\frac{2}{n}}\right),$$

which is $\leq \varepsilon$ for sufficiently large m since $\log(m)^{-\frac{n+3}{n+2}} \to 0$ and the exponent of ε is greater than 1.

□

4.3 Arbitrary Boolean Circuits and Deep Networks

In this section, we quote results from [2] to conclude the presentation of arbitrary Boolean circuits. Note that a different construction than the previous one is required to prove these results, which is presented in the appendix of [2].

Given that we can represent UAnds of any arity, we thus also can now concatenate ANDs of arities $1{:}n$, such that we can read off arbitrary Boolean functions from the result. Note that the width here scales exponentially in the depth L of the circuit.

Theorem 4.46. (Arbitrary Boolean circuits via 1-layer network, [2, Thm. 6]):
Let $\mathfrak{C}$ be any s-sparse Boolean circuit of width m and depth L. There exists a feature encoding $\Phi : \mathbb{R}^m \to \mathbb{R}^d$ and a single-layer neural network $\mathcal{M}(x) = \mathrm{ReLU}(\mathbf{W}_{\text{in}} x + b)$ of width $d = \tilde{\mathcal{O}}(\sqrt{m})$ such that $\mathcal{M} \circ \Phi$ ε-linearly represents $\mathfrak{C}$ on all s-sparse inputs with $\varepsilon \in \tilde{\mathcal{O}}\left(m^{-\frac{1}{3}}\right)$.

Proof Idea.
By Proposition 4.20, we can represent $\mathfrak{C}$ as a linear combination of And circuits of arities $1{:}n$ for some $n \in \mathbb{N}$. We then obtain the layer by stacking n UAnd layers of arities 1 to n. Using a linear combination of the read-offs of these UAnd layers, we can then represent $\mathfrak{C}$.

□

The previous result is only partially satisfactory, as we have exponential width in the depth L of the circuit. However, leveraging deeper networks, one can reduce this to sublinear width in m. However, for making networks deeper, one needs to be able to reduce the interference after each layer. This is possible using a dedicated error-reduction layer, as demonstrated in the next lemma.

Lemma 4.47. (Error reduction layer [2, Lem. 7]):
Assume that $m = \tilde{\mathcal{O}}\left(d^{\frac{3}{2}}\right)$, and c is some large polylog constant. Then for sufficiently small input interference $\varepsilon_{\text{in}} = \tilde{\mathcal{O}}\left(d^{-\frac{1}{2}}\right)$ there exists a one-layer MLP $\mathcal{M} : \mathbb{R}^d \to \mathbb{R}^d$ 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 $\varepsilon_{\text{out}} = \frac{\varepsilon_{\text{in}}}{c}$.

Alternating error-correction and UAnd layers, one can now represent arbitrary Boolean circuits with deep networks of sublinear width.

Theorem 4.48. (Arbitrary Boolean circuits with deep networks, [2, Thm. 8]):
Let $\mathfrak{C} : \{0,1\}^m \to \{0,1\}^m$ be a Boolean circuit of width m and of polynomial depth $L = \tilde{\mathcal{O}}(m^c)$. Then, there exists a feature encoding $\Phi : \mathbb{R}^m \to \mathbb{R}^d$ and a network $\mathcal{M}$ of width $d \in \tilde{\mathcal{O}}\left(m^{\frac{2}{3}} s^2\right)$ and depth $2L$ such that $\mathcal{M} \circ \Phi$ ε-linearly represents $\mathfrak{C}$ for all but a negligible fraction of inputs b on which $\mathfrak{C}$ is s-sparse.

4.4 Estimates and Empirical Results

4.4.1 Error Estimates for UAnd Networks

In the previous sections, we have given main results for representing Boolean functions in superposition using random neural network constructions. We formulated all results in terms of ε-linear representation with error ε that holds outside of negligible probability. However, this notion is qualitative and does not give explicit error estimates. In this section, we give estimates for error terms from the previous sections. For brevity, we only give main results here. Detailed derivations and accuracy discussions can be found in Appendix A.4.

The estimates here, including the full derivations in Appendix A.4 are original contributions from us.

Proposition 4.49. (Error estimates for UAnd with basis-aligned inputs):
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Let $s' := \|x\|_0 - x_1 - x_2$, i.e., the number of active features excluding the two relevant features x_1, x_2. Then, the expectation of the read-off is biased upwards by at most $s'p$:

$$\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x)] \lesssim x_1 x_2 + s'p.$$

The variance of the error is approximately bounded by

$$\mathrm{Var}(\varepsilon) = \mathrm{Var}(r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x) - x_1 x_2) \lesssim s'(dp)^{-1}.$$

Derivation.
See Proposition 1.10 and Proposition 1.11 in Appendix A.4. The expectation estimate assumes that the size of each respective neuron set $|\Gamma_2(i,j)|$ is strictly positive, which holds o.n.p. for the given growth conditions. For the variance bound, we also assume that $N = |\Gamma_2(1,2)|$ concentrates strongly around its expectation dp^2 and that N is approximately uncorrelated with the error sum. Furthermore, we assume $s'p \ll 1$.

□

Proposition 4.50. (Error estimates for UAnd with inputs in superposition):
Consider the UAnd construction with inputs in superposition from Corollary 4.35. The expectation of the error is approximately bounded as

$$\mathbb{E}[\varepsilon_{\mathrm{SP}}] \lesssim s'p + \frac{1}{2}(mp+2)^{\frac{1}{2}}\left(32sd^{-1}\log(m+1)\right)^{\frac{1}{2}} \in \mathcal{O}\left(sp + s^{\frac{1}{2}}m^{\frac{1}{2}}p^{\frac{1}{2}}d^{-\frac{1}{2}}\log(m)^{\frac{1}{2}}\right).$$

The variance of the error is approximately bounded as

$$\mathrm{Var}(\varepsilon_{\mathrm{SP}}) \lesssim s'(dp)^{-1} + 32s(2+mp^2)d^{-1}\log(m+1).$$

Derivation.
See Corollary 1.16 in Appendix A.4. The derivation combines the basis-aligned error with the error from reading off features from superposed inputs. We assume that the read-off noise from superposition is independent of the basis-aligned error. For the variance, we again assume strong concentration of N and that N is sufficiently large.

□

Note 4.51. (Width growth estimates for the Bernoulli UAnd with inputs in superposition):
For the Bernoulli construction, it is necessary, that dp^2 grows sufficiently fast to ensure that $N = |\Gamma_2|$ is sufficiently large, i.e., that we have sufficiently many rows to read-off each UAnd. This implies that we need to scale the probability at least as $p \in \Omega\left(d^{-\frac{1}{2}}\right)$. Thus, to reach an expected error of at most $\mathcal{O}(1)$, we conclude that we need a width growth of at least $d \in \Omega\left(m^{\frac{2}{3}}\right)$, which is consistent with Corollary 4.35. Notably, to control the variance, one would only need a width growth of $d \in \Omega\left(m^{\frac{1}{2}}\right)$. Thus, the error expectation is the limiting factor in this construction. We can locate the reason for this in the proof: The read-off error for each feature adds up due to the Bernoulli distribution having positive expectation. Thus, for neural networks dealing with superposed inputs, we expect approximately symmetric weight distributions to perform better. We will see that with a symmetric unbiased construction, we can reach better width scaling.

Proposition 4.52. (Error estimates for UAnd from general I.I.D. matrices):
Let $\boldsymbol{W} \in \mathbb{R}^{d \times m}$ be a random matrix with i.i.d. entries $W_{ij} \sim \mathcal{D}$ for some symmetric zero-mean distribution $\mathcal{D}$ with variance $0 < \sigma_{\mathcal{D}} < \infty$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Then, the estimator is unbiased, i.e., $\mathbb{E}[\varepsilon] = 0$. The variance of the error is approximately bounded by

$$\mathrm{Var}(\varepsilon) \lesssim s\left(\frac{\sigma_{\mathcal{D}}}{c_{\mathcal{D}}(s)}\right)^2 d^{-1},$$

where $c_{\mathcal{D}}(s) > 0$ is a constant dependent on the distribution $\mathcal{D}$ and sparsity s.

Derivation.
See Appendix A.4.2.4 in Appendix A.4. The unbiasedness follows from the symmetry of the weight distribution and the construction of the normalization factor η; the derivation is analogous to Lemma 4.40, assuming that the normalization factor is well-defined.

□

Proposition 4.53. (Error estimates for UAnd from general I.I.D. matrices with inputs in superposition):
Consider the UAnd construction from general I.I.D. matrices with inputs in superposition. The estimator is unbiased, i.e., $\mathbb{E}[\varepsilon_{\mathrm{SP}}] = 0$. The variance of the error is approximately bounded as

$$\mathrm{Var}(\varepsilon_{\mathrm{SP}}) \lesssim s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + 32sm\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-2}\log(m+1),$$

where $c'_{\mathcal{D}}(s)$ is a constant dependent on the distribution $\mathcal{D}$, the noise distribution and sparsity s.

Derivation.
See Corollary 1.21 in Appendix A.4. The result combines the variance estimate from the basis-aligned case with noise Proposition 1.20 with the feature read-off variance from Lemma 1.15. We assume that the noise contributions are uncorrelated with the signal read-off.

□

Note 4.54. (Width growth estimate for the symmetric I.I.D. case):
Since the estimator is unbiased, the width scaling is dominated by the variance. To ensure that the variance is bounded by a constant, we need the second term to be bounded, which implies $d^2 \in \Omega(m \log(m))$, i.e., $d \in \Omega\left(\sqrt{m \log(m)}\right)$. This is a significant improvement over the Bernoulli construction, which required $d \in \Omega\left(m^{\frac{2}{3}}\right)$ to control the error expectation and is consistent with the results that the authors from [2] obtain for their Rademacher-like construction.

4.4.2 Empirical Comparison

We now empirically compare the error of the Bernoulli construction, the Gaussian construction, and trained neural networks (TNN) on representing UAnd circuits in superposition.[12] We also compare a Rademacher construction, where weights are initialized i.i.d. from $\{-1, 1\}$, which is conceptually analogous to the Gaussian construction, but with discrete weights. For details, see Appendix A.4.3.

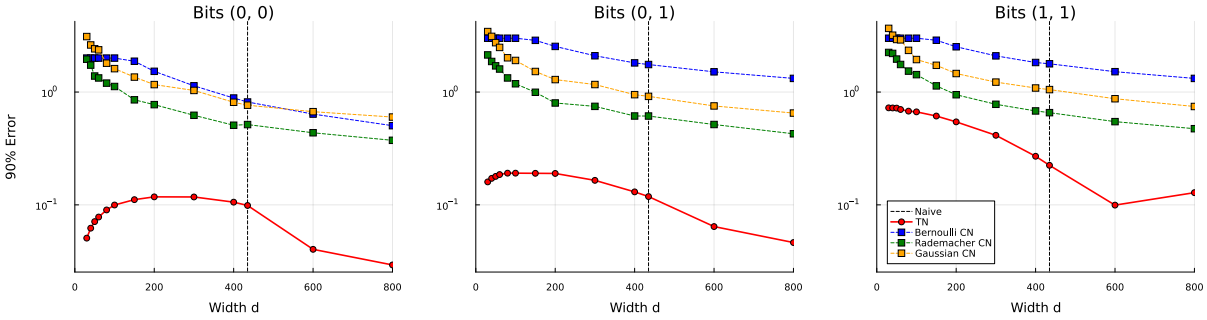


Figure 21: Comparison of the 90% error quantile for different constructions (CN) and trained networks (TN) against layer width. We compare the Bernoulli construction from Definition 4.23, the Gaussian construction from Theorem 4.39, and trained neural networks. We separate plots for different inputs to the And: Left: no input active. Center: one input active. Right: both inputs active. In all cases, the input sparsity is $s = 5$. We mark the naive computation bound as dashed line, which corresponds to the necessary width $\binom{m}{2}$ for computing each And separately.

We see that the trained networks outperform the random constructions significantly. For the constructions, we see that the Bernoulli construction has the highest error, followed by Gaussian and Rademacher constructions. We see a significant drop in error around the naive computation limit for the trained network. As expected for sparse training data, the network appears to first learn to represent the And for 0-outputs and then for 1-outputs as the width increases.

We are now interested in how the error of a construction for a specific sparsity changes when evaluated on other sparsities. For this, we use the trained networks from above, which were trained on sparsity $s = 5$, and the Gaussian constructions with read-off for sparsity $s = 5$. We then evaluate both on varying sparsities.

[12] The code for the empirical comparison can be found in `experimentsjl/error_estimates_trained.jl`.

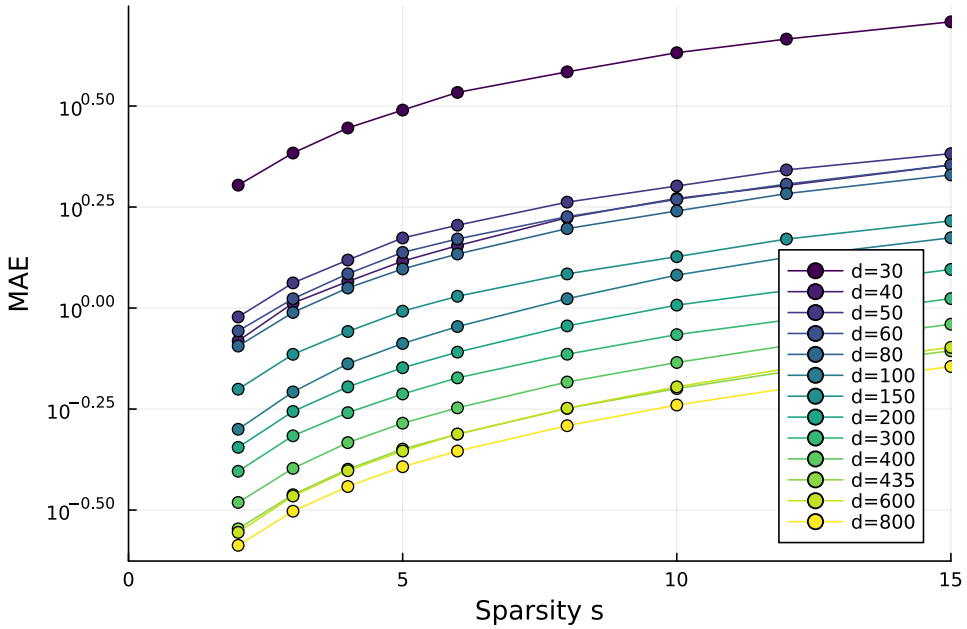


Figure 22: Mean Absolute Error (MAE) of the Gaussian construction for varying input sparsity s. Different lines correspond to different widths d.

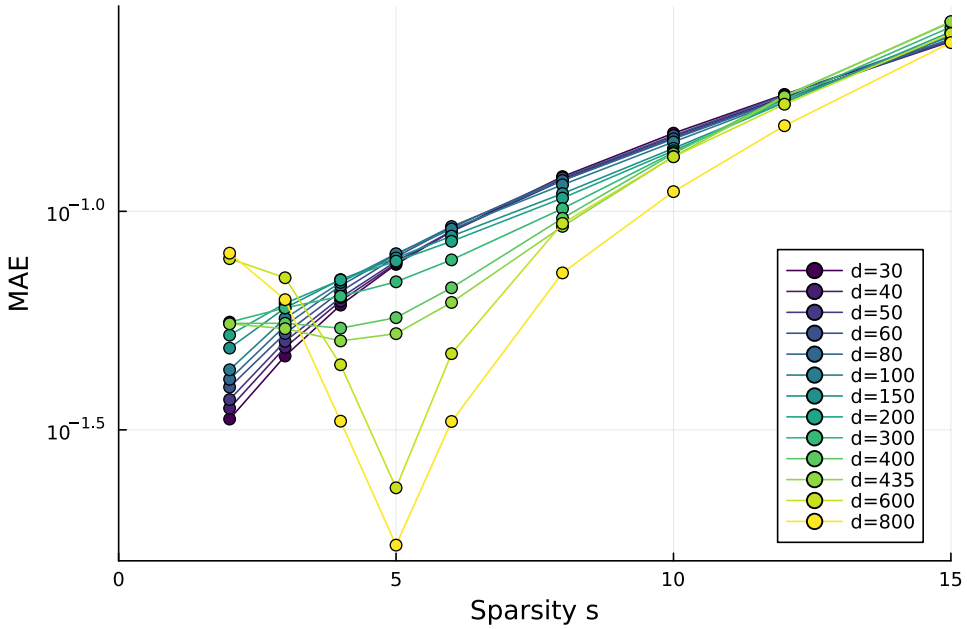


Figure 23: Mean Absolute Error (MAE) of trained neural networks for varying input sparsity s. The networks were trained on sparsity $s = 5$. Different lines correspond to different widths d.

Generally, we see a consistent rise of error with increasing sparsity for both Gaussian construction and trained networks, which qualitatively agrees with behavior of computation in superposition.

We see that large constructions beyond the naive computation limit fit to the training sparsity, which appears as significant drop for $s = 5$, while smaller constructions show more consistently rising behavior across sparsities. This suggests that networks specialize for sparsity seen during training and do not generalize to distant sparsities.

Notably, the low-width trained networks have very similar shapes as the Gaussian constructions, suggesting that they approximate the UAnd behavior via analogous mechanisms. Particularly, these

do not show a significant drop at the training sparsity, indicating that they do not specialize to the training sparsity.

4.5 Summary

In this chapter, we investigated random constructions of neural networks that can represent Boolean functions in superposition, following [2]. We first gave an overview of probabilistic tools used in this chapter, including negligible probability and concentration inequalities. The notion of negligible probability is useful, because it allows qualitative reasoning about events that occur with very high probability without tracking exact constants. Concentration inequalities, like the Hoeffding and Chernoff inequalities give us quantitative bounds on how random variables deviate from their expectations, which gives o.n.p. bounds e.g. on Binomials and functions of Gaussian variables, which appear in the error analyses. We then defined Boolean circuits and illustrated how arbitrary Boolean functions can be represented as linear combinations of And circuits of the respective arities. Elaborating the main text from [2], we then defined the Bernoulli UAnd network, which is a one-layer neural network with Bernoulli-initialized weights. Completing the sketch from [2], we showed that this ε-linearly represents the universal And circuit on s-sparse inputs with small error outside of negligible probability. We then gave theorems showing how chaining the basis-aligned Bernoulli UAnd with feature encoding gives a layer that operates on superposed inputs. Notably, this requires polynomial width in the number of inputs. Notably, instead of the width estimate $d \in \tilde{\mathcal{O}}\left(m^{\frac{1}{2}}\right)$ in the original paper, we only obtained $d \in \tilde{\mathcal{O}}\left(m^{\frac{2}{3}}\right)$. Next, we proved that the Basis-Aligned UAnd is also represented by Gaussian-initialized-MLPs, where we again extended the proof sketch in [2], additionally using the Gaussian Lipschitz concentration theorem to show concentration of the read-off around its expectation. Here, the main steps in the proof were to show that the expectation of the read-off matches the And operation and that the read-off is sufficiently concentrated around its expectation. We then extended the results to arbitrary arity Ands, with a Bernoulli construction with rescaled probability parameter and different bias. We quoted results for arbitrary Boolean circuits, first using single-layer networks with exponential width in the depth of the circuit, and finally deep networks with sublinear width in the number of inputs. Finally, we provided explicit error estimates for the Bernoulli and general i.i.d. constructions, deriving bounds for the expectation and variance of the error. We showed that symmetric weight distributions (like Gaussian or Rademacher) allow for better width scaling than the Bernoulli construction due to unbiased error expectation. We concluded with an empirical comparison of the theoretical constructions and trained neural networks. The results suggest that trained networks significantly improve upon the random constructions, however, at low widths they share qualitative characteristics with the Gaussian construction.

Chapter 5

Conclusions and Outlook

In this final chapter, we summarize the main results of this thesis and discuss potential directions for future research.

5.1 Conclusions

In this thesis, we have investigated representational and computational superposition in neural networks. We have mainly followed the influential blog post "Toy Models of Superposition" by Elhage et al. [1] and the article "Mathematical Models of Computation in Superposition" by Hanni et al. [2].

While a universal characterization of features is difficult, in this thesis we have mathematically considered features as maps from the input space to a feature space, which allows to define notions of representation and reducibility rigorously. As empirical results suggest that features in neural networks are not generally decomposable into one-dimensional features([34]), we have included adapted versions of definitions from [34] for multi-dimensional features, illustrating them with polygonal examples.

We fundamentally divided between representational superposition—the ability to store and retrieve features—and computational superposition, which is about directly computing with features in superposition.

From a purely theoretical point of view, representational superposition is mathematically possible with exponentially many features. Particularly, the number of dimensions provided from the JL-lemma rises as $d \in \Omega(\log(m)\varepsilon^{-2}s^2)$ for m features, sparsity s, and maximum read-off error ε (Lemma 4.31).

In Chapter 3, we have seen that a two-layer autoencoder architecture provides a model for the emergence of representational superposition. Concretely, we have shown that superposition is never advantageous for the linear autoencoder. However, for non-linear activations such as ReLU, superposition becomes practically advantageous. Considering high-sparsity first order approximations, we have seen that practical representational superposition balances feature benefit and interference.

In homogeneous settings, we have seen the emergence of polygon weight configurations. We have proven that these are stationary points of the zero-bias loss. Furthermore, our stability analysis suggests that complex polygon solutions become unstable for larger numbers of features (around $n_f \geq 9$), where embedded lower-complexity solutions have lower loss. This theoretically explains the empirical observation that models tend to learn simpler geometric structures like hexagons rather than high-degree polygons.

We have also seen that representational superposition in the toy model is robust across many ReLU-like activation functions and optimizer settings. Notably, it occurs in both coupled and uncoupled architectures, with uncoupled models—which are more representative of real-world networks—often learning features more reliably.

In Chapter 4, we investigated random constructions of layers performing computation on features in superposition. We focused on the Universal And circuit, which serves as a basis for arbitrary Boolean computation. Using probabilistic tools such as concentration inequalities, we formulated

and proved statements about the Bernoulli UAnd and Gaussian constructions from [2]. Specifically, for basis-aligned inputs, a single-layer network with Bernoulli-initialized weights can represent the Universal And circuit with width polylogarithmic in the number of features ($d \in \Omega(\log(m^5)\varepsilon^{-2})$), which is similar to the bound on representational superposition. We also showed that Gaussian-initialized networks can achieve similar polylogarithmic width bounds when operating on basis-aligned inputs.

However, when operating directly on superposed inputs (by chaining with feature encoding), the width requirements increase. Our analysis of the Bernoulli construction yielded a polynomial width bound of $d \in \tilde{\mathcal{O}}\left(m^{\frac{2}{3}}\right)$, which is slightly looser than the $\tilde{\mathcal{O}}\left(m^{\frac{1}{2}}\right)$ bound suggested in [2]. We saw that this is likely due to the bias present in the Bernoulli construction, which leads to higher interference when operating on superposed inputs. We obtained an analogous result by deriving explicit expectation and variance estimates in Section 4.4. These suggest that symmetric, unbiased distributions (such as Gaussian or Rademacher weights) allow for better width scaling of $d \in \tilde{\mathcal{O}}\left(\sqrt{m\log(m)}\right)$, consistent with the tighter bounds.

Empirically, we found that trained neural networks achieve lower errors at comparable widths as the random constructions. Notably, at low widths, trained networks exhibit behavior qualitatively similar to the i.i.d. random constructions, suggesting reliance on similar mechanisms.

Based on these results, it appears that under strict sparsity assumptions and assuming real features as base model of computation, superposition is a representation strategy that allows for both representational and computational efficiency. When superposition is present as computational strategy, the constructions in Chapter 4 suggest that we can expect a polynomial sublinear scaling of the network width in the number of features.

5.2 Limitations

Our analysis relies on several simplifying assumptions to make the mathematical treatment feasible.

First, regarding the data distribution, we assumed strict sparsity (bounded by a fixed s) and strictly Boolean features. In reality, sparsity is likely more accurately modeled by softer models (e.g., power-law distributed activations) and features often carry continuous information (e.g., magnitude representing confidence or strength). Second, regarding the model architecture, we primarily focused on simple, shallow feed-forward networks. While we touched upon deep networks, the rigorous bounds for computation are strongest for single-layer constructions. Third, the asymptotic results assume sufficiently large network widths to apply concentration inequalities, which may not fully capture the behavior of smaller, finite-width networks used in practice.

5.3 Outlook

Several of these limitations point to directions for future research.

A direct extension would be to generalize the computational models to non-Boolean features and softer sparsity constraints, decreasing the gap to realistic activations. Furthermore, extending the analysis to more complex architectures appears promising. This includes analyzing computation in deep networks more rigorously and, importantly, investigating superposition in residual architectures, where we have different error propagation dynamics that may allow more efficient computation. Another interesting aspect is the relation to other representation strategies, such as compositionality, such as described in [44], which suggests that superposition and compositionality trade off against each other.

Great practical interest lies in developing computational methods to decompose neural representations into meaningful components, which is strongly related to representation strategies.

Examples of current methods include [45], where interpretable features are extracted from trained networks and a computational graph is built to represent their interactions.

While this thesis focused on the feature-centric view of superposition, the difficulty of defining 'features' suggests exploring alternative interpretability frameworks. Currently, more general network decomposition methods like Stochastic Parameter Decomposition (SPD) [35] are being developed, which appear promising for obtaining a more complete empirical understanding of representation strategies in neural networks.

Appendix

A.1 Notation

A.1.1 Conventions

When referring to functions, we sometimes add argument names although we refer to the function itself, e.g., "the function $f(x)$" instead of "the function f". This is to emphasize the parametric form of the function (here, that it is usually used on x-values) and is common practice in engineering and physics.

In cases where we have independent random variables $X_1, ..., X_n$, we often write $\mathbb{E}_{X_1}[f(X_1, X_2, ..., X_n)]$ to denote the expectation over X_1 only. I.e., this is the same as writing $\mathbb{E}\left[f(X_1, X_2, ..., X_n) \mid X_2, ..., X_n\right]$. When having multi-line derivations with inequalities or equalities, the operator at the beginning of the line applies to the previous expression. For example, in

$$\begin{aligned} f(x) &= g(x) \\ &\leq g(x) + 1 \\ &= h(x), \end{aligned}$$

the first line states $f(x) = g(x)$, the second line states $g(x) \leq g(x) + 1$, and the last line states $g(x) + 1 = h(x)$.

A.1.2 Notation Overview

Table 2: Overview of mathematical notation used in this thesis.

Notation	Description
$i{:}j$	Range of integers, defined as $i{:}j \coloneqq \{i, i+1, ..., j\}$
u	Vector
$\boldsymbol{A} \in \mathbb{R}^{d_1 \times d_2}$	Matrix A (bold face) with d_1 rows and d_2 columns
A_{ij}	Entry in row i and column j of matrix $\boldsymbol{A}$
A_i	Row i of matrix $\boldsymbol{A}$ as vector, with $A_i \in \mathbb{R}^{d_2}$
A^j	Column j of matrix $\boldsymbol{A}$ as vector, with $A^j \in \mathbb{R}^{d_1}$
$\hat{u}$	A unit vector. If u is a vector, defined as $\hat{u} \coloneqq \frac{u}{\|u\|}$
$u \cdot v$	Standard scalar product
$V \odot U$	Componentwise product of tensors
$\mathbf{1}$	Identity matrix
$(x_i)_i$	Vector or sequence constructed from entries x_i
$\boldsymbol{A} \coloneqq (A_{ij})_{ij}$	Matrix constructed from entries A_{ij}
$\mathbf{1}_n$	'Eye' matrix, defined as $\mathbf{1}_n \coloneqq ([i = j][i \leq n])_{ij}$. Has n ones in the first n entries of the diagonal and zeros elsewhere, representing the identity on a subspace. Shape defined by context.
$\mathrm{diag}(u)$	Diagonal matrix with vector u on the diagonal

Notation	Description
e_i	Standard basis vector
$\mathrm{colspace}(\boldsymbol{A})$	Column space of matrix A
$\binom{u}{v}$	Vertical concatenation of vectors $u \in \mathbb{R}^{d_1}$ and $v \in \mathbb{R}^{d_2}$, resulting in $\binom{u}{v} \in \mathbb{R}^{d_1+d_2}$
$\mathrm{cvxhull}(S)$	Convex hull of a set S, the smallest convex set containing all points in S
$\|x\|_0$	0-Pseudonorm, defined as $\|x\|_0 \coloneqq \lvert\{x_i \mid x_i \neq 0\}\rvert$. Number of nonzero elements.
$\|x\|_2$	Euclidean norm (2-norm), defined as $\|x\|_2 \coloneqq \left(\sum_i x_i^2\right)^{\frac{1}{2}}$
$\|\boldsymbol{A}\|_F$	Frobenius norm, defined as $\|\boldsymbol{A}\|_F \coloneqq \left(\sum_{i,j}(a_{ij}^2)\right)^{\frac{1}{2}}$
$\|x\|_\infty$	Infinity norm, defined as $\|x\|_\infty \coloneqq \max_i \lvert x_i\rvert$
$\mathbb{P}(\mathcal{A})$	Probability of event $\mathcal{A}$, or general probability measure
$\mathbb{E}[X]$	Expected value of random variable X
$\mathrm{Var}(X)$	Variance of random variable X, defined as $\mathrm{Var}(X) \coloneqq \mathbb{E}\big[(X - \mathbb{E}[X])^2\big]$
$\mathrm{supp}(X)$	Support of a random variable, smallest closed set A such that $\mathbb{P}(X \in A) = 1$
$\mathcal{A}$	An event, i.e., a measurable set in respective σ-algebra. Denoted by calligraphic letters.
$\complement(A)$	Complement of event or set A
$\lvert A\rvert$	Number of elements in set A
$\mathcal{U}[a, b]$	Uniform distribution on interval $[a, b]$, with density $f(x) \coloneqq \frac{1}{b-a}(a \leq x \leq b)$
$\phi(x)$	Standard Gaussian probability density function (PDF), defined as $\varphi(x) \coloneqq \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$
$\Phi(x)$	Standard Gaussian cumulative distribution function (CDF), defined as $\Phi(x) \coloneqq \left(\frac{1}{\sqrt{2\pi}}\right) \int_{-\infty}^{x} \exp\left(-\frac{t^2}{2}\right) dt$
$\mathcal{N}(\mu, \sigma^2)$	Gaussian (normal) distribution with mean μ and variance σ^2. Density: $f(x) \coloneqq \sigma^{-1}\phi\left(\frac{x-\mu}{\sigma}\right)$
$\mathcal{N}_+(\mu, \sigma^2)$	Half-normal or half-Gaussian distribution. We have $\lvert X\rvert \sim \mathcal{N}_+$ for $X \sim \mathcal{N}(0, \sigma^2)$
$\texttt{Ber}(p)$	Bernoulli distribution with success probability p. We have $\mathbb{P}(X = 1) = p$ and $X \in \{0, 1\}$ for $X \sim \texttt{Ber}(p)$.
$\texttt{Bin}(n, p)$	Binomial distribution. Represents the sum of n i.i.d. $\texttt{Ber}(p)$ random variables.
$\mathcal{O}(\cdot)$	Big-O notation (asymptotic upper bound). We have $f(n) \in \mathcal{O}(g(n)) :\Leftrightarrow \limsup \frac{f(n)}{g(n)} < \infty$
$\tilde{\mathcal{O}}(\cdot)$	Soft-O notation ($\mathcal{O}$ ignoring logarithmic factors). We have $f(n) \in \tilde{\mathcal{O}}(g(n))$ if $f(n) \in \mathcal{O}\big(\log(n)^k g(n)\big)$ for some $k \geq 0$.
$\Theta(\cdot)$	Theta notation (tight asymptotic bound). We have $f(n) \in \Theta(g(n)) :\Leftrightarrow f(n) \in \mathcal{O}(g(n))$ and $g(n) \in \mathcal{O}(f(n))$.
$\Omega(\cdot)$	Omega notation (asymptotic lower bound). We have $f(n) \in \Omega(g(n)) :\Leftrightarrow g(n) \in \mathcal{O}(f(n))$

Notation	Description
$\tilde{\Omega}(\cdot)$	Soft-Omega notation (asymptotic lower bound ignoring logs)
1_{expr}	Iverson bracket: 1 if expression is true, 0 otherwise
$(x > y)x$	Example of a boolean expression, interpreted as 1 if true and 0 if false
$\mathrm{ReLU}(x)$	Rectified Linear Unit (ReLU) activation function, defined as $\mathrm{ReLU}(x) \coloneqq x(x > 0) = \max(x, 0)$
$(x)_+$	Shorthand for ReLU
$\mathrm{LeakyReLU}(x)$	Leaky ReLU, defined as $\mathrm{LeakyReLU}(x) \coloneqq x(x > 0) + \alpha x(x \leq 0)$ for small $\alpha > 0$
$\mathrm{ELU}_{\alpha(x)}$	Exponential Linear Unit, defined as $\mathrm{ELU}_{\alpha(x)} \coloneqq x(x > 0) + \alpha(\exp(x) - 1)(x \leq 0)$
$\mathrm{GELU}(x)$	Gaussian Error Linear Unit, defined as $\mathrm{GELU}(x) \coloneqq x\Phi(x)$ where Φ is the standard Gaussian CDF
$\mathrm{SiLU}(x)$	Sigmoid Linear Unit (Swish), defined as $\mathrm{SiLU}(x) \coloneqq x\,\sigma(x)$
$\sigma(x)$	Logistic (sigmoid) function, defined as $\sigma(x) \coloneqq (1 + \exp(-x))^{-1}$
$\mathrm{sgn}(x)$	Sign function. Returns -1 for negative, 0 for zero, and 1 for positive inputs.
$\mathrm{clamp}(x, [a, b])$	Clamp function, defined as $\mathrm{clamp}(x, [a, b]) \coloneqq \min(\max(x, a), b)$
m	Number of features
n_h	Number of hidden units
$\hat{X}$	Reconstructed or estimated version of X
$f(x) \propto g(x)$	f is proportional to g, i.e., there exists a constant c such that $f(x) = c \cdot g(x)$ for all x
$f \equiv c, X \equiv c$	Emphasize that a function f is constant or that a random variable X is constant
$[f(x)]_{x_0}^{x_1}$	Evaluation bracket, defined as $[f(x)]_{x_0}^{x_1} \coloneqq f(x_1) - f(x_0)$
$X \approx Y$	Heuristic approximation: X is approximated by Y
$X \lesssim Y$	Heuristic inequality: X is approximately less than or equal to Y
$x \ll y$	Much less than (heuristic). x is significantly smaller than y.

Table 3: Common abbreviations.

Abbreviation	Meaning
iff	if and only if
s.t.	such that
w.r.t.	with respect to
w.l.o.g.	without loss of generality
o.n.p.	outside of negligible probability
a.s.	Almost sure(-ly), with probability 1
a.e.	Almost everywhere, up to measure 0

Abbreviation	Meaning
BA	Basis-Aligned
NE	Individual Noise Error
SP	Superposition
WN	With Noise

A.2 Additional Theorems

Note 1.1. (Mills ratio for standard Gaussian variable):
Let $Z \sim \mathcal{N}(0,1)$ be a standard normal random variable. Then, for any $z > 0$, we have the upper bound on the tail probability

$$\mathbb{P}(Z > z) \leq \frac{\phi(z)}{z},$$

where ϕ is the PDF of the standard normal distribution.

Proof.
Using integration by parts, we have

$$\begin{aligned}
\mathbb{P}(Z > z) &= \int_z^\infty (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{t^2}{2}\right) \, \mathrm{dt} \\
&\leq (2\pi)^{-\frac{1}{2}} \int_z^\infty \underbrace{\left(\frac{t}{z}\right)}_{\geq 1} \exp\left(-\frac{t^2}{2}\right) \, \mathrm{dt} \\
&= (2\pi)^{-\frac{1}{2}} \left(\frac{1}{z}\right) \left[-\exp\left(-\frac{t^2}{2}\right)\right]_{t=z}^{t=\infty} \\
&= \frac{\phi(z)}{z}.
\end{aligned}$$

□

Lemma 1.2. (Rectified Gaussian Expectation):
Let $X \sim \mathcal{N}(\mu, \sigma^2)$ be a Gaussian random variable. Then, we have

$$\mathbb{E}[\mathrm{ReLU}(X)] = \mu\Phi\left(\frac{\mu}{\sigma}\right) + \sigma\varphi\left(\frac{\mu}{\sigma}\right),$$

where Φ is the cumulative distribution function and φ the probability density function of the standard normal distribution.

Proof.
First, for a standard normal random variable $Z \sim \mathcal{N}(0,1)$, we have

$$\mathbb{E}\left[Z 1_{Z > z_0}\right] = \int_{z_0}^\infty z\varphi(z) \, \mathrm{dz} = \varphi(z_0).$$

Write $X = \sigma Z + \mu$ with $Z \sim \mathcal{N}(0,1)$. Then, the expectation of the rectified Gaussian decomposes into two parts:

$$\begin{aligned}\mathbb{E}[\mathrm{ReLU}(X)] &= \mathbb{E}\big[(\sigma Z+\mu)1_{\sigma Z+\mu>0}\big] \\ &= \sigma\ \mathbb{E}\Big[Z1_{Z>-\frac{\mu}{\sigma}}\Big] + \mu\ \mathbb{E}\Big[1_{Z>-\frac{\mu}{\sigma}}\Big] \\ &= \sigma\ \mathbb{E}\Big[Z1_{Z>-\frac{\mu}{\sigma}}\Big] + \mu\ \mathbb{P}\Big(Z>-\frac{\mu}{\sigma}\Big) \\ &= \sigma\varphi\Big(\frac{\mu}{\sigma}\Big) + \mu\Phi\Big(\frac{\mu}{\sigma}\Big).\end{aligned}$$

□

Lemma 1.3. (Independence of And functions):
The set of 2^a And functions $\Big\{\bigwedge_{i\in I} x_i \mid I \subseteq 1{:}a\Big\}$ is linearly independent in the vector space of all Boolean-to-real functions $\{0,1\}^a \to \mathbb{R}$.

Proof.
To see linear independence, consider the 'evaluation' matrix $M \in \{0,1\}^{2^a\times 2^a}$, where columns correspond to the And functions indexed by sets $I \subseteq 1{:}a$ and rows correspond to the 2^a possible inputs $x \in \{0,1\}^a$. We identify each input x with the set of its active indices $J = \big\{j \mid x_j = 1\big\}$. We now define the entry $M_{J,I}$ to be the value of the And function for set I evaluated on input J, i.e.,

$$M_{J,I} := \left[\bigwedge_{i\in I} x_i\right]^{x=1_J} = 1_{I\subseteq J},$$

We order the rows and columns by the size of the sets $|J|$ and $|I|$ in ascending order and lexicographically for sets of the same size. If $|I| \geq |J|$ and $I \neq J$, then I cannot be a subset of J, so $M_{J,I} = 0$. Thus, the matrix is lower triangular. Furthermore, for $I = J$, we have $I \subseteq J$, so the diagonal entries are all 1.

Since M is a lower triangular matrix with non-zero diagonal entries, it has full rank. Therefore, its columns, corresponding to the respective And functions, are linearly independent.

□

A.3 Notes on Mathematical Models of Superposition

In this appendix, we provide commentary on specific differences between the theorems presented in this thesis and [2].

Note 1.4. (Omission of case distinction and resulting different bounds in Corollary 4.29):
The authors distinguish one additional case in the original proof of [2, Thm. 1]:

Let $x_1 = x_2 = 0$. Then, the interference terms have value at most s and there are at most

$$\sum_{k_1 \neq k_2, x_{k_1} = x_{k_2} = 1} |\Gamma(1,2,k_1,k_2)| \in \Theta(s^2 \log(m)^8 d^{-1}) \qquad \text{o.n.p.}$$

such terms. Thus the error is o.n.p.

$$\frac{1}{|\Gamma(1,2)|} s^2 \Theta(\log(m)^8 d^{-1}) = \Theta(s^3 \log(m)^4 d^{-1}),$$

giving the seemingly worse bound $s^3 \log(m)^4 d^{-1}$.

However, this case is already covered by the second case. In particular, note that the number of interference terms is

$$\begin{aligned} &\sum_{k_1 \neq k_2, x_{k_1} = x_{k_2} = 1} |\Gamma(1,2,k_1,k_2)| \\ &= \sum_{k_1} \sum_{k_2 \neq k_1, x_{k_2} = 1} |\Gamma(1,2,k_1) \cap \Gamma(k_2)| \\ &\leq \sum_{k_1} |\Gamma(1,2,k_1)|, \end{aligned}$$

which is the number of interference terms as in the second case. Thus, the additional bound $s^3 \log(m)^4 d^{-1}$ is an artificial result of the case distinction. We therefore have a slightly improved overall bound without the need for a case distinction.

A.4 Error Estimates for U-And in Superposition

Here, we derive heuristic estimates for constructions in Chapter 4. Specifically, we analyze the variance and expectation of the error when computing functions in superposition. This provides an alternative view on the o.n.p. error bounds derived in Chapter 4 and gives quantitative estimates for the error. Notably, we can also derive scaling for the width given a target error and compare to the bounds obtained in Chapter 4. Note that these estimates are heuristic and rely on several approximations to limit complexity.

A.4.1 Preliminaries

We first summarize some useful properties of the variance operation.

Lemma 1.5. (Properties of variance):
Let X, Y be random variables, and $a, b \in \mathbb{R}$ constants. Then, the following properties hold:

(i) $\operatorname{Var}(aX) = a^2 \operatorname{Var}(X)$

(ii) $\operatorname{Var}(X+Y) = \operatorname{Var}(X) + \operatorname{Var}(Y) + 2\operatorname{Cov}(X,Y)$. If X and Y are uncorrelated, we have $\operatorname{Cov}(X,Y) = 0$ and thus $\operatorname{Var}(X+Y) = \operatorname{Var}(X) + \operatorname{Var}(Y)$.

(iii) $\operatorname{Var}(X+a) = \operatorname{Var}(X)$

(iv) $\operatorname{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$

(v) Let f be a Lipschitz function with Lipschitz constant L. Then,

$$\operatorname{Var}(f(X)) \leq L^2 \operatorname{Var}(X)$$

(vi) Let $\overline{X}, \overline{Y}$ be independent and have zero mean. Then,

$$\operatorname{Var}\left(\overline{X}\,\overline{Y}\right) = \operatorname{Var}\left(\overline{X}\right)\operatorname{Var}\left(\overline{Y}\right) = \mathbb{E}\left[\overline{X}^2\right]\,\mathbb{E}\left[\overline{Y}^2\right]$$

(vii) Let X, Y be independent. Then,

$$\operatorname{Var}(XY) = \operatorname{Var}(X)\operatorname{Var}(Y) + \operatorname{Var}(X)(\mathbb{E}[Y])^2 + \operatorname{Var}(Y)(\mathbb{E}[X])^2$$

Lemma 1.6. (Variances for common distributions):
- For Bernoulli $X \sim \texttt{Ber}(p)$, we have $\operatorname{Var}(X) = p(1-p)$
- For binomial $X \sim \texttt{Bin}(n,p)$, we have $\operatorname{Var}(X) = np(1-p)$
- For uniform $X \sim \mathcal{U}(a,b)$, we have $\operatorname{Var}(X) = \frac{1}{12}(b-a)^2$
- For Rademacher $X \sim \texttt{Rad}$, we have $\operatorname{Var}(X) = 1$

Note 1.7. (Notation for approximations):
In order to limit complexity, we use heuristic approximations to derive estimates.
- $X \approx Y$ means that X is approximated by Y.
- $X \lesssim Y$ means that X is approximately less than or equal to Y.

See also Appendix A.1 for notation.

Note 1.8. (Heuristic Taylor approximations for small variance):
Let X be a random variable and $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. Then, for small $\mathrm{Var}(X)$, we approximately have

$$\mathbb{E}[f(X)] \approx f(\mathbb{E}[X])$$

and

$$\mathrm{Var}(f(X)) \approx (f'(\mathbb{E}[X]))^2 \,\mathrm{Var}(X).$$

Derivation.
Approximating f around the mean $\mu = \mathbb{E}[X]$ using a second-order Taylor expansion, we have

$$\begin{aligned}
\mathbb{E}[f(X)] &\approx \mathbb{E}\left[f(\mu) + f'(\mu)(X-\mu) + \frac{1}{2}f''(\mu)(X-\mu)^2\right] \\
&= f(\mu) + f'(\mu)\underbrace{\mathbb{E}[X-\mu]}_{=0} + \frac{1}{2}f''(\mu)\underbrace{\mathbb{E}[(X-\mu)^2]}_{=\mathrm{Var}(X)} \\
&= f(\mathbb{E}[X]) + \frac{1}{2}f''(\mathbb{E}[X])\,\mathrm{Var}(X).
\end{aligned}$$

For small variance, the second term vanishes, yielding $\mathbb{E}[f(X)] \approx f(\mathbb{E}[X])$.

For the variance, using a first-order Taylor expansion:

$$\begin{aligned}
\mathrm{Var}(f(X)) &\approx \mathrm{Var}(f(\mu) + f'(\mu)(X-\mu)) \\
&= (f'(\mu))^2 \,\mathrm{Var}(X-\mu) \\
&= (f'(\mathbb{E}[X]))^2 \,\mathrm{Var}(X).
\end{aligned}$$

□

In order to convert variance estimates into concrete bounds, we use the Chebyshev inequality:

Proposition 1.9. (Chebyshev Inequality):
Let X be a random variable with finite expectation $\mathbb{E}[X]$ and finite non-zero standard deviation σ_X. Then, for any $\alpha > 0$, we have that the probability that X deviates from its expectation by at least α standard deviations is bounded inverse-quadratically in α, i.e.,

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq \alpha\sigma_X) \leq \alpha^{-2}.$$

A.4.2 UAnd with Basis-Aligned Inputs

Expectation

We first analyze the expectation of the error for the UAnd construction with basis-aligned inputs from Corollary 4.29. Notably, this construction is a biased estimator of the UAnd function—we have $\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\mathtt{Ber}}) \geq x_1 x_2]$.

Proposition 1.10. (Expectation estimate for UAnd with basis-aligned inputs):
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Let $s' := \|x\|_0 - x_1 - x_2$, i.e., the number of active features excluding the two relevant features x_1, x_2. Then, we have

$$\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x)] \lesssim x_1 x_2 + s'p.$$

Derivation.
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Let $s' := \|x\|_0 - x_1 - x_2$, i.e., the number of active features excluding the two relevant features x_1, x_2. We analyze w.l.o.g. the expectation of the read-off for the first UAnd, $x_1 x_2$.

Let $N = |\Gamma_2(1,2)|$. Assume $N > 0$ and let w.l.o.g. $1 \in \Gamma_2(1,2)$. As all weights are i.i.d., we have

$$\begin{aligned}\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x)] &= \mathbb{E}\left[\frac{1}{N} \sum_{i \in \Gamma_2(1,2)} \mathrm{ReLU}\left(-1 + x_1 + x_2 + \sum_{k \geq 3} W_{\texttt{Ber}\,ik} x_k\right)\right] \\ &= \mathbb{E}\left[\mathrm{ReLU}\left(-1 + x_1 + x_2 + \sum_{k \geq 3} W_{\texttt{Ber}\,1k} x_k\right)\right].\end{aligned}$$

We have two cases:

- For $x_1 + x_2 \geq 1$, the ReLU is always active, thus we can drop it. We thus obtain

$$\begin{aligned}\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x)] &= \mathbb{E}\left(-1 + x_1 + x_2 + \sum_{k \geq 3} W_{\texttt{Ber}\,1k} x_k\right) \\ &= x_1 x_2 + s'p.\end{aligned}$$

- For $x_1 = x_2 = 0$, we can upper bound the inner term by omitting -1, which yields

$$\begin{aligned}\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x)] &\leq \mathbb{E}\left(0 + 0 + \sum_{k \geq 3} W_{\texttt{Ber}\,1k} x_k\right) \\ &= x_1 x_2 + s'p.\end{aligned}$$

Thus, in both cases, we have $\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x)] \leq x_1 x_2 + s'p$. I.e., the expectation of the read-off is biased upwards by at most $s'p$.

□

Variance Estimation
We first analyze the variance of the error for the UAnd construction with basis-aligned inputs from Corollary 4.29.

Proposition 1.11. (Variance estimate for UAnd with basis-aligned inputs):
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Let $s' := \|x\|_0 - x_1 - x_2$, i.e., the number of active features excluding the two And features x_1, x_2. Then, the variance of the error is approximately bounded by

$$\mathrm{Var}(\varepsilon) = \mathrm{Var}(r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} x) - x_1 x_2) \lesssim s'(dp)^{-1}.$$

Derivation.
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Let $s' \coloneqq \|x\|_0 - x_1 - x_2$, i.e., the number of active features excluding the two relevant features x_1, x_2. We analyze w.l.o.g. the variance of the read-off for the first UAnd, $x_1 x_2$. Define $N \coloneqq |\Gamma_2(1,2)|$. We assume $N > 0$, which holds o.n.p. for sufficiently large dp^2. The variance of the error is given by

$$\begin{aligned}
\operatorname{Var}(\varepsilon) &= \operatorname{Var}(r \cdot \operatorname{ReLU}(\mathbf{W}_{\texttt{Ber}} x) - x_1 x_2) \\
&= \operatorname{Var}\left(\frac{1}{N} \sum_i \left(\operatorname{ReLU}\left(-1 + x_1 + x_2 + \sum_{k \geq 3} W_{\texttt{Ber}_{ik}} x_k\right) - x_1 x_2\right)\right) \\
&= \operatorname{Var}\left(\frac{1}{N} Y\right)
\end{aligned}$$

where we defined $Y \coloneqq \sum_i \left(\operatorname{ReLU}\left(-1 + x_1 + x_2 + \sum_{k \geq 3} W_{\texttt{Ber}_{ik}} x_k\right) - x_1 x_2\right)$ as the sum term.

We now assume that N and the error Y are approximately uncorrelated. Thus, the variance of the product approximately decomposes as

$$\begin{aligned}
\operatorname{Var}(\varepsilon) &= \operatorname{Var}\left(\frac{1}{N} Y\right) && \downarrow \text{Product variance} \\
&\approx \operatorname{Var}\left(\frac{1}{N}\right)\left(\mathbb{E}[Y]^2 + \operatorname{Var}(Y)\right) + \operatorname{Var}(Y)\left(\mathbb{E}\left[\frac{1}{N}\right]\right)^2
\end{aligned}$$

We estimate the individual terms. Note that N follows a Binomial distribution $\texttt{Bin}(d, p^2)$.

Estimating $\operatorname{Var}\left(\frac{1}{N}\right)$: Assuming strong concentration of N, by Note 1.8 with $f(x) \coloneqq x^{-1}$, we have

$$\operatorname{Var}(f(x)) \approx \left(f'(\mathbb{E}[X])\right)^2 \operatorname{Var}(X) = \left(\mathbb{E}[N]\right)^{-4} \operatorname{Var}(N) = \left(dp^2\right)^{-4} dp^2(1-p^2) = \left(dp^2\right)^{-3}(1-p^2).$$

Estimating $\mathbb{E}[Y]$: We have $\mathbb{E}[Y] \approx \mathbb{E}[N] s' p = dp^2 s' p = ds' p^3$.

Estimating $\mathbb{E}\left[\frac{1}{N}\right]$: Assuming strong concentration of N, we have $\mathbb{E}\left[\frac{1}{N}\right] \approx \left(\mathbb{E}[N]\right)^{-1} = \left(dp^2\right)^{-1}$.

Estimating $\operatorname{Var}(Y)$:

$$\begin{aligned}
\operatorname{Var}(Y) &\approx \mathbb{E}[N] \operatorname{Var}\left(\operatorname{ReLU}\left(-1 + x_1 + x_2 + \sum_{k \geq 3} W_{\texttt{Ber}_{ik}} x_k\right)\right) && \downarrow \text{Sum of variances} \\
&\leq dp^2 s' p(1-p) = ds' p^3 (1-p).
\end{aligned}$$

Thus, we have

$$\begin{aligned}
&\operatorname{Var}(\varepsilon) \\
&\approx \operatorname{Var}\left(\frac{1}{N}\right)\left(\mathbb{E}[Y]^2 + \operatorname{Var}(Y)\right) + \operatorname{Var}(Y)\left(\mathbb{E}\left[\frac{1}{N}\right]\right)^2 && \downarrow \text{Substitute estimates} \\
&\leq \left(dp^2\right)^{-3}(1-p^2)\left(\left(ds' p^3\right)^2 + ds' p^3(1-p)\right) + ds' p^3(1-p)\left(dp^2\right)^{-2} \\
&= d^{-3} p^{-6}(1-p^2)(d^2 s'^2 p^6 + ds' p^3(1-p)) + ds' p^3(1-p) d^{-2} p^{-4} && \downarrow \text{Drop } \mathcal{O}\left(\left(dp^2\right)^{-1}\right) \text{term} \\
&\approx d^{-1} s'^2 (1-p^2) + s'(dp)^{-1}(1-p) && \downarrow \text{Assume } s'p \ll 1 \quad (1.1.1) \\
&\approx s'(dp)^{-1}.
\end{aligned}$$

Notably, the term $s'(dp)^{-1}$ asymptotically dominates for small p.

□

Notably, the final simplification can also be obtained by assuming $N \approx dp^2$ constant and computing the resulting variance directly.

Note 1.12. (Validity of Approximation):
The approximation above assumes that the number of active features $|\Gamma(1,2)|$ concentrate strongly around its expectation dp as well as that it is non-zero. Thus, similarly to the proof of Corollary 4.29, this is only valid if dp is sufficiently large.

UAnd with Inputs in Superposition
We now consider the U-And construction with inputs in superposition, which is constructed by chaining the Bernoulli U-And from Definition 4.23 with the symmetrized random feature encoding from Note 4.32. The negligible bound analysis can be found in Corollary 4.35. Notably, the expectation-variance analysis gives similar scaling laws as the negligible bound analysis.

We first analyze the expectation of the error for the UAnd construction with noised inputs from Section 4.2.1.

Proposition 1.13. (Expectation estimate for UAnd for incoming noise):
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s and let $\tilde{x} = x + \Delta\mathrm{X}$ be the input vector with incoming noise $\Delta\mathrm{X} \in \mathbb{R}^m$. Let $s' \coloneqq \|x\|_0 - x_1 - x_2$. Let the noise $\Delta\mathrm{X}_j$ be i.i.d. with $\mathbb{E}\left[\Delta\mathrm{X}_j\right] = 0$ for all j with variance $\mathrm{Var}\left(\Delta\mathrm{X}_j\right) = \sigma_{\text{in}}^2$.

We assume $N \geq 1$, which holds o.n.p. for sufficiently large dp.

Then, we approximately have

$$\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}}\tilde{x})] \lesssim x_1 x_2 + s'p + \frac{1}{2}(mp+2)^{\frac{1}{2}}\sigma_{\text{in}}.$$

Derivation.
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s and let $\tilde{x} = x + \Delta\mathrm{X}$ be the input vector with incoming noise $\Delta\mathrm{X} \in \mathbb{R}^m$. Let the noise $\Delta\mathrm{X}_j$ be i.i.d. with $\mathbb{E}\left[\Delta\mathrm{X}_j\right] = 0$ for all j with variance $\mathrm{Var}\left(\Delta\mathrm{X}_j\right) = \sigma_{\text{in}}^2$.

We assume $N \geq 1$, which holds o.n.p. for sufficiently large dp.

Then, we have

$$
\begin{aligned}
&\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}}\tilde{x})] \\
&= \mathbb{E}\left[\frac{1}{N}\sum_{i\in\Gamma_2(1,2)} \mathrm{ReLU}\left(-1+\tilde{x}_1+\tilde{x}_2+\sum_{k\geq 3} W_{\texttt{Ber}\,ik}\tilde{x}_k\right)\right] && \downarrow \text{Symmetry of rows} \\
&= \mathbb{E}\left[\mathrm{ReLU}\left(-1+\tilde{x}_1+\tilde{x}_2+\sum_{k\geq 3} W_{\texttt{Ber}\,1k}\tilde{x}_k\right)\right] && \downarrow \text{Subadditivity of ReLU.} \\
&\leq \mathbb{E}\left[\mathrm{ReLU}\left(-1+x_1+x_2+\sum_{k\geq 3} W_{\texttt{Ber}\,1k}x_k\right)\right] \\
&+\mathbb{E}\left[\mathrm{ReLU}\left(\Delta \mathrm{X}_1+\Delta \mathrm{X}_2+\sum_{k\geq 3} W_{\texttt{Ber}\,1k}\Delta \mathrm{X}_k\right)\right].
\end{aligned}
$$

The inner term of the first expectation is the same before in the basis-aligned case. For the second expectation, we use that for symmetric random variables Z, we have $\mathbb{E}[\mathrm{ReLU}(Z)] = \frac{1}{2}\,\mathbb{E}[|Z|]$. Thus, we have

$$
\begin{aligned}
&\mathbb{E}[r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}}\tilde{x})] \\
&\leq x_1x_2 + s'p + \frac{1}{2}\,\mathbb{E}\left[\left|\Delta \mathrm{X}_1+\Delta \mathrm{X}_2+\sum_{k\geq 3} W_{\texttt{Ber}\,1k}\Delta \mathrm{X}_k\right|\right] && \downarrow \mathbb{E}[|Z|] \leq (\mathbb{E}[Z^2])^{\frac{1}{2}} \\
&\leq x_1x_2 + s'p + \frac{1}{2}\left(\mathrm{Var}(\Delta \mathrm{X}_1)+\mathrm{Var}(\Delta \mathrm{X}_2)+\sum_{k\geq 3}\mathrm{Var}\left(W_{\texttt{Ber}\,1k}\Delta \mathrm{X}_k\right)\right)^{\frac{1}{2}} \\
&= x_1x_2 + s'p + \frac{1}{2}(2\sigma_{\text{in}}^2 + (m-2)p\sigma_{\text{in}}^2)^{\frac{1}{2}} \\
&\leq x_1x_2 + s'p + \frac{1}{2}(mp+2)^{\frac{1}{2}}\sigma_{\text{in}}.
\end{aligned}
$$

□

We now analyze the variance of the UAnd construction when the inputs are in superposition.

First, as in Section 4.2.1, we first analyze the variance of the Bernoulli construction when we have noised inputs and then combine this with the variance from reading off features from superposed inputs.

Proposition 1.14. (Variance estimate for UAnd with incoming noise):
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s and let $\tilde{x} = x + \Delta \mathrm{X}$ be the input vector with incoming noise $\Delta \mathrm{X} \in \mathbb{R}^m$. Let $s' := \|x\|_0 - x_1 - x_2$. Let the noise $\Delta \mathrm{X}_j$ be i.i.d. with $\mathbb{E}[\Delta \mathrm{X}_j] = 0$ and $\mathrm{Var}(\Delta \mathrm{X}_j) = \sigma_{\text{in}}^2$ for all j.

Then, the variance of the error is approximately bounded by

$$\mathrm{Var}(\varepsilon) \approx \sigma_{\text{BA}}^2 + (2+mp^2)\sigma_{\text{in}}^2 \lesssim s'(dp)^{-1} + (2+mp^2)\sigma_{\text{in}}^2.$$

Derivation.
Let $\mathbf{W}_{\texttt{Ber}}$ be the Bernoulli weight matrix from Definition 4.23 with dimensions $d \times m$. Let $\tilde{x} = x + \Delta \mathrm{X}$ be the input vector with incoming noise $\Delta \mathrm{X} \in \mathbb{R}^m$. Let the noise $\Delta \mathrm{X}_j$ be i.i.d. with $\mathbb{E}[\Delta \mathrm{X}_j] =$

0 and $\mathrm{Var}(\Delta \mathrm{X}_j) = \sigma_{\text{in}}^2$ for all j. We analyze w.l.o.g. the variance of the read-off for the first And $x_1 x_2$.

Assume the average case $N = |\Gamma_2(1,2)| \approx dp^2$.

Then, we have

$$\begin{aligned}
&\mathrm{Var}(r \cdot \mathrm{ReLU}(\mathbf{W}_{\texttt{Ber}} \tilde{x})) \\
&= \mathrm{Var}\left(\frac{1}{N} \sum_i \mathrm{ReLU}\left(-1 + \sum_k W_{\texttt{Ber}_{ik}} \tilde{x}_k \right) \right) \\
&\approx \sigma_{\text{BA}}^2 + \mathrm{Var}\left(\frac{1}{N} \sum_i \sum_k W_{\texttt{Ber}_{ik}} \Delta \mathrm{X}_k \right) && \downarrow \text{Additivity of variance} \\
&= \sigma_{\text{BA}}^2 + N^{-2} \sum_k \mathrm{Var}\left(\Delta \mathrm{X}_k \sum_i W_{\texttt{Ber}_{ik}} \right) && \downarrow \text{Independence of } \Delta \mathrm{X}_k \\
&= \sigma_{\text{BA}}^2 + N^{-2} \sigma_{\text{in}}^2 \sum_k \mathbb{E}\left[\left(\sum_i W_{\texttt{Ber}_{ik}} \right)^2 \right].
\end{aligned}$$

We split the sum over k into $k \in \{1,2\}$ and $k \geq 3$. For $k \in \{1,2\}$, we have $W_{\texttt{Ber}_{ik}} = 1$ for all $i \in \Gamma_2(1,2)$, thus $\sum_i W_{\texttt{Ber}_{ik}} = N$. For $k \geq 3$, $\sum_i W_{\texttt{Ber}_{ik}} \sim \texttt{Bin}(N,p)$, thus $\mathbb{E}\left[\left(\sum_i W_{\texttt{Ber}_{ik}} \right)^2 \right] \approx (Np)^2$. Thus,

$$\begin{aligned}
\mathrm{Var}(\varepsilon) &\approx \sigma_{\text{BA}}^2 + N^{-2} \sigma_{\text{in}}^2 (2N^2 + m(Np)^2) \\
&= \sigma_{\text{BA}}^2 + \sigma_{\text{in}}^2 (2 + mp^2).
\end{aligned}$$

I.e., we have an extra variance term of approximately $\sigma_{\text{NE}} = \sigma_{\text{in}}^2 (2 + mp^2)$.

□

Now, we derive the variance from reading off features from superposed inputs.

Lemma 1.15. (Variance of feature read-off):
Let $\mathbf{\Phi} \in \mathbb{R}^{d \times m}$ be the symmetrized random feature encoding from Note 4.32. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Then, the variance of the read-off for any feature i is bounded by

$$\sigma_{\text{in}}^2 = \mathrm{Var}\left(\phi_i \cdot \sum_j x_j \phi_j \right) \leq 32 s d^{-1} \log(m+1).$$

Derivation.
Let $\mathbf{\Phi} \in \mathbb{R}^{d \times m}$ be the symmetrized random feature encoding. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Note that all read-off interferences are uncorrelated:

$$\mathrm{Cov}(\phi_i \cdot \phi_j, \phi_k \cdot \phi_l) = 0 \qquad \text{if } \{i,j\} \neq \{k,l\}$$

This holds as in Note 4.32, each feature vector ϕ_i is multiplied by an independent random sign.

We now derive a bound for the maximum absolute scalar product $\varepsilon' := \max_{i \neq j} |\phi_i \cdot \phi_j|$. From Corollary 2.9, we have that the scalar products are bounded by $\varepsilon' = 2\varepsilon$ with high probability if

$$d \geq 4\log(m+1)\left(\frac{1}{2}\varepsilon^2 + \frac{1}{3}\varepsilon^3\right)^{-1} \geq 4\log(m+1)\left(\frac{1}{2}\varepsilon^2\right)^{-1} = 8\log(m+1)\varepsilon^{-2} = 32\log(m+1)(\varepsilon')^{-2}.$$

Rearranging for ε', we obtain

$$(\varepsilon')^2 \leq 32d^{-1}\log(m+1).$$

As $\phi_i \cdot \phi_j$ for $i \neq j$ is zero mean and bounded by ε', we have that the total read-off variance for x_i is bounded by

$$\begin{aligned}\sigma_{\text{in}}^2 = \operatorname{Var}\left(\phi_i \cdot \sum_j x_j\phi_j\right) &= \sum_{j,j\neq i} x_j \operatorname{Var}(\phi_i \cdot \phi_j) \\ &\leq s(\varepsilon')^2 \\ &\leq 32sd^{-1}\log(m+1).\end{aligned}$$

□

Corollary 1.16. (Error estimates for U-And with inputs in superposition):
Consider the U-And construction with inputs in superposition from Corollary 4.35. The expectation of the error is approximately bounded as

$$\mathbb{E}[\varepsilon_{\text{SP}}] \lesssim s'p + \frac{1}{2}(mp+2)^{\frac{1}{2}}(32sd^{-1}\log(m+1))^{\frac{1}{2}} \in \mathcal{O}\left(sp + s^{\frac{1}{2}}m^{\frac{1}{2}}p^{\frac{1}{2}}d^{-\frac{1}{2}}\log(m)^{\frac{1}{2}}\right).$$

The variance of the error is approximately bounded as

$$\operatorname{Var}(\varepsilon_{\text{SP}}) \lesssim s'(dp)^{-1} + 32s(2+mp^2)d^{-1}\log(m+1).$$

Derivation.
From Lemma 1.15, we have that the variance from reading off features from superposed inputs is bounded by $\sigma_{\text{in}}^2 \leq 32sd^{-1}\log(m+1)$.

Expectation: For the expectation, from Proposition 1.13, we have

$$\begin{aligned}\mathbb{E}[\varepsilon_{\text{SP}}] &\lesssim s'p + \frac{1}{2}(mp+2)^{\frac{1}{2}}\sigma_{\text{in}} \\ &\leq s'p + \frac{1}{2}(mp+2)^{\frac{1}{2}}(32sd^{-1}\log(m+1))^{\frac{1}{2}} \in \mathcal{O}\left(sp + s^{\frac{1}{2}}m^{\frac{1}{2}}p^{\frac{1}{2}}d^{-\frac{1}{2}}\log(m)^{\frac{1}{2}}\right).\end{aligned}$$

Variance: For the variance, combining Proposition 1.14 and the read-off variance from Lemma 1.15, we obtain the total variance estimate for the UAnd with inputs in superposition:

$$\begin{aligned}\operatorname{Var}(\varepsilon) &\approx s'(dp)^{-1} + (2+mp^2)\sigma_{\text{in}}^2 \\ &\leq s'(dp)^{-1} + (2+mp^2)32sd^{-1}\log(m+1).\end{aligned}$$

□

Reading off U-And from General I.I.D. Matrices
We can read off the U-And function from general i.i.d. matrices when the entries are symmetrically distributed.

As in the Gaussian construction, we first define the unnormalized read-off and then normalize it, such that the expectation matches the U-And function. Notably, this construction generalizes the Gaussian construction in Lemma 4.40.

Definition 1.17. (U-And read-off from general I.I.D. matrices):
Let $\boldsymbol{W} \in \mathbb{R}^{d \times m}$ be a random matrix with i.i.d. entries $W_{ij} \sim \mathcal{D}$ for some symmetric zero-mean distribution $\mathcal{D}$ with variance $0 < \sigma_{\mathcal{D}} < \infty$. The unnormalized read-off vector is defined such that it counts rows positive where both weight entries have the same sign and negative otherwise:

$$\tilde{r}_i := \operatorname{sgn}(W_{i1}) \operatorname{sgn}(W_{i2})$$

We then define the normalization factor

$$\eta := \mathbb{E}[\tilde{r} \cdot \operatorname{ReLU}(\boldsymbol{W}x)]$$

for a fixed Boolean input vector $x \in \{0,1\}^m$ with sparsity s and $x_1 = x_2 = 1$. The normalized read-off vector is then defined as

$$r := \eta^{-1} \tilde{r}$$

Proposition 1.18. (Variance estimate for UAnd from general I.I.D. matrices):
Let $\boldsymbol{W} \in \mathbb{R}^{d \times m}$ be a random matrix with i.i.d. entries $W_{ij} \sim \mathcal{D}$ for some symmetric zero-mean distribution $\mathcal{D}$ with variance $\sigma_{\mathcal{D}} < \infty$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. Let $c_{\mathcal{D}}(s) := \frac{\eta}{d}$ be the per-neuron normalization factor. Then, the variance of the error is bounded by

$$\operatorname{Var}(\varepsilon) \leq s \left(\frac{\sigma_{\mathcal{D}}}{c_{\mathcal{D}}(s)} \right)^2 d^{-1}.$$

Derivation.
Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s. First, consider the normalization factor η. We have

$$\begin{aligned}
\eta &= \sum_i \mathbb{E}\left[\operatorname{sgn}(W_{i1}) \operatorname{sgn}(W_{i2}) \operatorname{ReLU}\left(W_{i1} + W_{i2} + \sum_{k \geq 3} W_{ik} x_k \right) \right] \\
&= d \; \mathbb{E}\left[\operatorname{sgn}(W_{11}) \operatorname{sgn}(W_{12}) \operatorname{ReLU}\left(W_{11} + W_{12} + \sum_{k \geq 3} W_{1k} x_k \right) \right] \\
&=: d c_{\mathcal{D}}(s),
\end{aligned}$$

with some constant $c_{\mathcal{D}}(s)$ dependent on the distribution $\mathcal{D}$ and sparsity s. In the following, we will assume that $c_{\mathcal{D}}(s) > 0$ for all relevant sparsities s. Now, we analyze the variance of the error $\varepsilon = r \cdot \operatorname{ReLU}(\boldsymbol{W}x) - x_1 x_2$. We have

$$\begin{aligned}
\operatorname{Var}(\varepsilon) &= \operatorname{Var}\left(\eta^{-1} \sum_i \operatorname{sgn}(W_{i1}) \operatorname{sgn}(W_{i2}) \operatorname{ReLU}\left(W_{i1} + W_{i2} + \sum_{k \geq 3} W_{ik} x_k \right) - x_1 x_2 \right) \\
&= \eta^{-2} d \operatorname{Var}(Y)
\end{aligned}$$

where we defined $Y := \operatorname{sgn}(W_{11}) \operatorname{sgn}(W_{12}) \operatorname{ReLU}\left(W_{11} + W_{12} + \sum_{k \geq 3} W_{1k} x_k \right)$. We now estimate $\operatorname{Var}(Y)$. Define $S := \operatorname{sgn}(W_{11}) \operatorname{sgn}(W_{12})$ and $Z := W_{11} + W_{12} + \sum_{k \geq 3} W_{1k} x_k$. We have

$$\begin{aligned}
\operatorname{Var}(Y) &\leq \operatorname{Var}(S \cdot \operatorname{ReLU}(Z)) && \downarrow \text{Variance is smaller than second moment} \\
&\leq \mathbb{E}\left[\underbrace{S^2}_{=1} \operatorname{ReLU}(Z)^2\right] && \downarrow \operatorname{ReLU}(z)^2 \leq z^2 \\
&\leq \mathbb{E}[Z^2] && \downarrow Z \text{ is zero mean} \\
&= \operatorname{Var}(Z) \\
&= x_1\sigma_{\mathcal{D}}^2 + x_2\sigma_{\mathcal{D}}^2 + \sum_{k\geq 3} x_k \sigma_{\mathcal{D}}^2 \\
&= s\sigma_{\mathcal{D}}^2.
\end{aligned}$$

Thus, the total variance is bounded by

$$\begin{aligned}
\operatorname{Var}(\varepsilon) &= \eta^{-2} d \operatorname{Var}(Y) \\
&\leq \left(d c_{\mathcal{D}}(s)\right)^{-2} d s \sigma_{\mathcal{D}}^2 \\
&= s\left(\frac{\sigma_{\mathcal{D}}}{c_{\mathcal{D}}(s)}\right)^2 d^{-1}.
\end{aligned}$$

□

Remark 1.19.
We see that the variance approximately linearly decreases with the layer width d and depends on the property of the distribution $\mathcal{D}$ via the ratio $\frac{\sigma_{\mathcal{D}}}{c_{\mathcal{D}}(s)}$.

Variance with Incoming Noise

We now analyze the variance of the U-And read-off from general i.i.d. matrices when there is incoming noise on the input. As before, let $\boldsymbol{W} \in \mathbb{R}^{d\times m}$ be a random matrix with i.i.d. entries $W_{ij} \sim \mathcal{D}$ for some symmetric zero-mean distribution $\mathcal{D}$ with variance $\sigma_{\mathcal{D}} < \infty$. Notably, we need to adjust the normalization factor to account for the incoming noise. We define

$$\eta' \coloneqq \mathbb{E}[\tilde{r} \cdot \operatorname{ReLU}(\boldsymbol{W}\tilde{x})]$$

for a fixed Boolean input vector $x \in \{0,1\}^m$ with sparsity s and $x_1 = x_2 = 1$ and noised input $\tilde{x} = x + \Delta\mathrm{X}$ with random, symmetric noise $\Delta\mathrm{X} \in \mathbb{R}^m$. The normalization constant is then given by $\eta' = d c'_{\mathcal{D}}(s)$ for some constant $c'_{\mathcal{D}}(s)$ dependent on the distribution $\mathcal{D}$, sparsity s, and noise distribution.

Proposition 1.20. (Variance estimate for UAnd from general I.I.D. matrices with incoming noise):
Let $\boldsymbol{W} \in \mathbb{R}^{d\times m}$ be a random matrix with i.i.d. entries $W_{ij} \sim \mathcal{D}$ for some symmetric zero-mean distribution $\mathcal{D}$ with variance $\sigma_{\mathcal{D}} < \infty$. Let $x \in \{0,1\}^m$ be a fixed Boolean input vector with sparsity s and let $\tilde{x} = x + \Delta\mathrm{X}$ be the input vector with incoming noise $\Delta\mathrm{X} \in \mathbb{R}^m$. Let the noise $\Delta\mathrm{X}_j$ be i.i.d. with $\mathbb{E}\left[\Delta\mathrm{X}_j\right] = 0$ and $\operatorname{Var}\left(\Delta\mathrm{X}_j\right) = \sigma_{\text{in}}^2$ for all j.

Then, the variance of the error is approximately bounded by

$$\operatorname{Var}(\varepsilon) \lesssim s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + m d^{-1} \sigma_{\text{in}}^2 \left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2.$$

Derivation.
Let $\tilde{x} = x + \Delta\mathrm{X}$ be the input vector with incoming noise $\Delta\mathrm{X} \in \mathbb{R}^m$. Let the noise $\Delta\mathrm{X}_j$ be i.i.d. with $E\big[\Delta\mathrm{X}_j\big] = 0$ and $\mathrm{Var}\big(\Delta\mathrm{X}_j\big) = \sigma_{\text{in}}^2$ for all j. We analyze w.l.o.g. the variance of the read-off for the first And $x_1 x_2$.

Notably, differently from the Bernoulli case, we have that the noise contributions for $j \notin \{1, 2\}$ in each row are uncorrelated, as the multiplication with the symmetric entries W_{ij} decorrelates them. Only for $j \in \{1, 2\}$, the sign contributions introduce correlation.

Similarly to before, we decompose the variance into the variance of one individual row contribution:

$$\begin{aligned}
\mathrm{Var}(\varepsilon) &= \mathrm{Var}(r \cdot \mathrm{ReLU}(\boldsymbol{W}\tilde{x})) \\
&= \mathrm{Var}\left(\eta^{-1} \sum_i \mathrm{sgn}(W_{i1})\,\mathrm{sgn}(W_{i2})\,\mathrm{ReLU}\left(W_{i1}\tilde{x}_1 + W_{i2}\tilde{x}_2 + \sum_{k\geq 3} W_{ik}\tilde{x}_k\right)\right) \\
&= \eta^{-2} \sum_i \mathrm{Var}(Y_i) + 2\sum_{i<j} \mathrm{Cov}\big(Y_i, Y_j\big) \\
&= \eta^{-2} d\,\mathrm{Var}(Y_1)
\end{aligned}$$

with $Y_i \coloneqq \mathrm{sgn}(W_{i1})\,\mathrm{sgn}(W_{i2})\,\mathrm{ReLU}\Big(W_{i1}\tilde{x}_1 + W_{i2}\tilde{x}_2 + \sum_{k\geq 3} W_{ik}\tilde{x}_k\Big)$.

We now estimate $\mathrm{Var}(Y_1)$. Define signs as before as $S \coloneqq \mathrm{sgn}(W_{11})\,\mathrm{sgn}(W_{12})$ and $\tilde{Z} \coloneqq W_{11}\tilde{x}_1 + W_{12}\tilde{x}_2 + \sum_{k\geq 3} W_{1k}\tilde{x}_k$. We have

$$\begin{aligned}
\mathrm{Var}(Y_1) &\leq \mathrm{Var}\big(S \cdot \mathrm{ReLU}\big(\tilde{Z}\big)\big) && \downarrow \text{Variance bounded by second moment} \\
&\leq \mathbb{E}\left[\underbrace{S^2}_{=1}\mathrm{ReLU}\big(\tilde{Z}\big)^2\right] && \downarrow \mathrm{ReLU}(z)^2 \leq z^2 \\
&\leq \mathbb{E}\big[\tilde{Z}^2\big] && \downarrow \tilde{Z} \text{ is zero mean} \\
&= \mathrm{Var}\big(\tilde{Z}\big) && \downarrow \text{Uncorrelated components} \\
&= \mathrm{Var}\left(\sum_j W_{1j}x_j\right) + \mathrm{Var}\left(\sum_j W_{1j}\Delta\mathrm{X}_j\right) \\
&= s\sigma_{\mathcal{D}}^2 + \sigma_{\mathcal{D}}^2\sigma_{\text{in}}^2 m.
\end{aligned}$$

Thus, the total variance is approximately bounded by

$$\begin{aligned}
\mathrm{Var}(\varepsilon) &\lesssim \eta^{-2} d\,\mathrm{Var}(Y_1) \\
&= \big(d c'_{\mathcal{D}}(s)\big)^{-2} d\big(s\sigma_{\mathcal{D}}^2 + \sigma_{\mathcal{D}}^2\sigma_{\text{in}}^2 m\big) \\
&= s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + m d^{-1}\sigma_{\text{in}}^2\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2.
\end{aligned}$$

□

We now again can construct a layer which operates on inputs in superposition by multiplying the weight matrix with a symmetrized random feature encoding from Note 4.32. Combining the variance from reading off features from superposed inputs from Lemma 1.15 with the variance estimate above, we obtain the following result.

Corollary 1.21. (Error estimates for U-And from general I.I.D. matrices with inputs in superposition):
Consider the U-And construction from general I.I.D. matrices with inputs in superposition. The estimator is unbiased, i.e., $\mathbb{E}[\varepsilon_{\mathrm{SP}}] = 0$. The variance of the error is approximately bounded as

$$\mathrm{Var}(\varepsilon_{\mathrm{SP}}) \lesssim s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + 32sm\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-2}\log(m+1).$$

Derivation.
From the symmetric construction, we have zero-mean error due to the balanced number of positive and negative contributions in the read-off, analogously to Lemma 4.40.

From Lemma 1.15, we have that the variance from reading off features from superposed inputs is bounded by $\sigma_{\mathrm{in}}^2 \leq 32sd^{-1}\log(m+1)$.

Substituting this into the variance estimate from Appendix A.4.2.5, we obtain:

$$\begin{aligned}
\mathrm{Var}(\varepsilon) &\lesssim s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + md^{-1}\sigma_{\mathrm{in}}^2\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 \\
&\leq s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + md^{-1}(32sd^{-1}\log(m+1))\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 \\
&= s\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-1} + 32sm\left(\frac{\sigma_{\mathcal{D}}}{c'_{\mathcal{D}}(s)}\right)^2 d^{-2}\log(m+1).
\end{aligned}$$

□

A.4.3 Empirical Evaluation

Calculations

We want to empirically verify how the superposition constructions compare to each other. We consider several constructions:

- The Bernoulli construction from Definition 4.23 with probability parameter $p = \log(m)^2 d^{-\frac{1}{2}}$ from [2].
- The general i.i.d. construction from Appendix A.4.2.4, using
  - A standard normal distribution.
  - A Rademacher distribution.

Notably, we have both biased and unbiased estimators here. The Bernoulli construction is biased, while the general i.i.d. constructions are unbiased. In order to compare the error bounds obtained from biased an unbiased estimators, we use the Chebyshev inequality to obtain high-probability bounds from the variance and expectation estimates.

Computation Details

For the Rademacher distribution, we compute the constant $c_{\mathcal{D}}(s)$ exactly. Here, we have

$$\begin{aligned}
c_{\mathcal{D}}(s) &= \mathbb{E}\left[\mathrm{sgn}(W_1)\,\mathrm{sgn}(W_2)\,\mathrm{ReLU}\left(W_1 + W_2 + \sum_{k=3}^{s} W_k\right)\right] \\
&= \frac{1}{4}(h(2) - 2h(0) + h(-2))
\end{aligned}$$

where we use the expectation of the ReLU of a shifted variable, $h(\mu) = \mathbb{E}[\mathrm{ReLU}(\mu + Z)]$ with $Z = \sum_{k=3}^{s} W_k$. Since $W_k \in \{-1, 1\}$, Z follows a shifted Binomial distribution $Z = 2K - (s-2)$ with $K \sim \mathtt{Bin}\big(s-2, \frac{1}{2}\big)$. Thus,

$$h(\mu) = \sum_{k=0}^{s-2} \mathrm{ReLU}(\mu + 2k - (s-2)) \binom{s-2}{k} 2^{-(s-2)}.$$

For the Gaussian distribution, we compute $c_{\mathcal{D}}(s)$ via numerical integration. Let $Z = \sum_{k=3}^{s} W_k$. Since $W_k \sim \mathcal{N}(0, 1)$, we have $Z \sim \mathcal{N}(0, s-2)$. Let $h(\mu) = \mathbb{E}[\mathrm{ReLU}(\mu + Z)]$. From Lemma 1.2, we have the analytical form of the ReLU expectation:

$$h(\mu) = \mu\Phi\left(\frac{\mu}{\sigma_Z}\right) + \sigma_Z \phi\left(\frac{\mu}{\sigma_Z}\right)$$

where $\sigma_Z = \sqrt{s-2}$. Then,

$$c_{\mathcal{D}}(s) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{sgn}(w_1 w_2) h(w_1 + w_2) \phi(w_1) \phi(w_2) \, \mathrm{d}w_1 \, \mathrm{d}w_2.$$

Accuracy

We compare the analytical Chebyshev bounds with sampled statistics for three constructions: Bernoulli, Rademacher, and Gaussian.[13]

The sampling parameters are $N_{\text{nets}} = 5$ networks and $N_{\text{vecs}} = 100$ vectors per network. The model parameters are $m \in [10, 50]$ and $s = 5$.

A.4.4 Error Statistics vs. Width

In order to see how the estimates compare to sampled statistics, we plot the error expectation and standard deviation against the width d. Note, that for the unbiased estimators, we do not include the expectation estimate, which is zero. Since the input bits to the U-And were important aspects in the derivations, we plot different lines for the cases where both input bits are active, one input bit is active, and no input bits are active.

Note that we use (1.1.1) for the analytical standard deviation estimate of the Bernoulli construction and Corollary 1.21 for the Rademacher and Gaussian constructions.

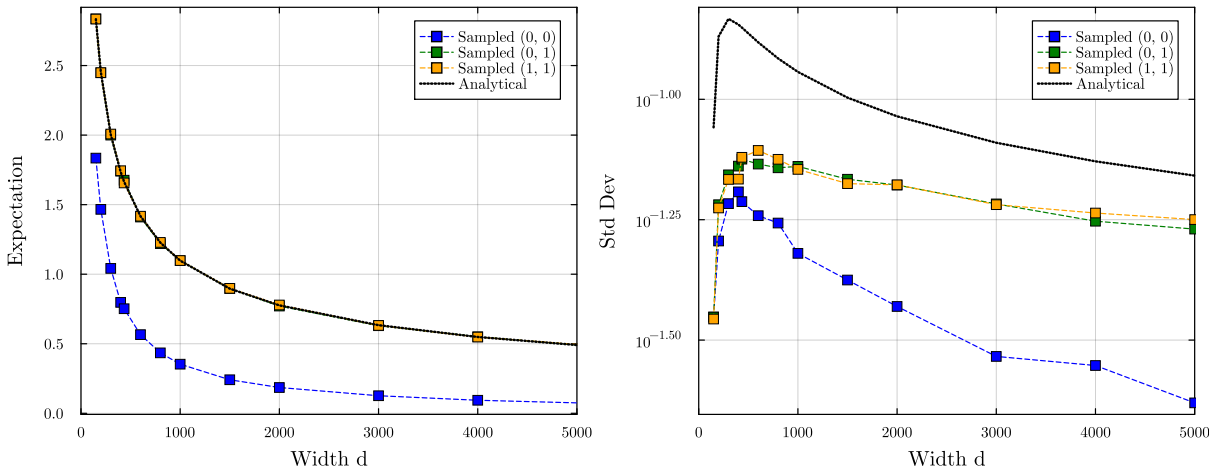


Figure 24: Bernoulli Construction: Error Expectation and Std Dev vs Width.

[13] The code for the calculations can be found in `experimentsjl/error_estimates.jl`. For the trained models, see `experimentsjl/error_estimates_trained.jl`.

We see that the analytical estimates of the Bernoulli expectation very closely matches the analytical estimate for all cases but the none-active case. In the standard deviation, we see that the analytical estimate is significantly looser than the sampled statistics, but still captures the shape and qualitative behavior of the decrease well. Again, we see that the none-active case deviates more from the analytical estimate.

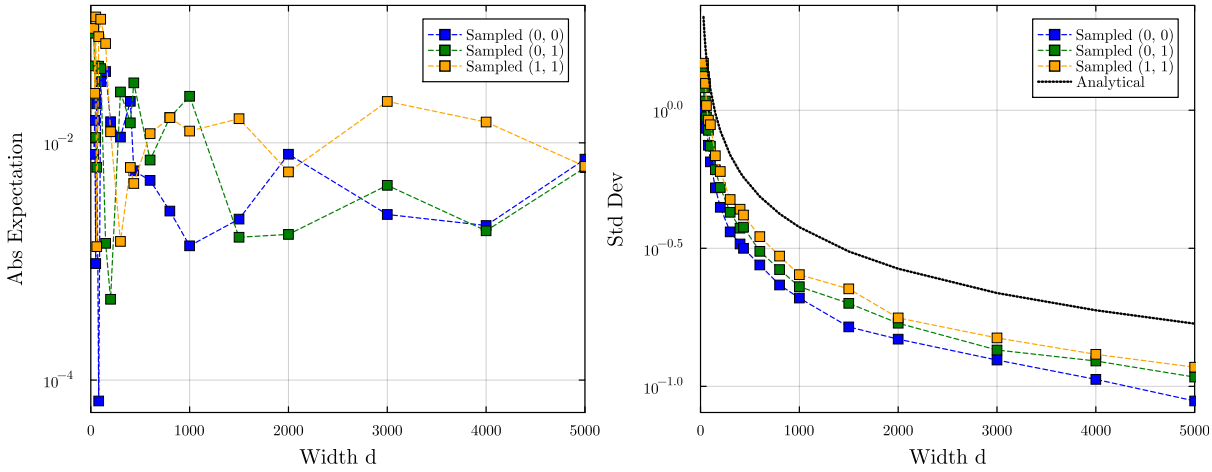


Figure 25: Rademacher Construction: Error Expectation and Std Dev vs Width.

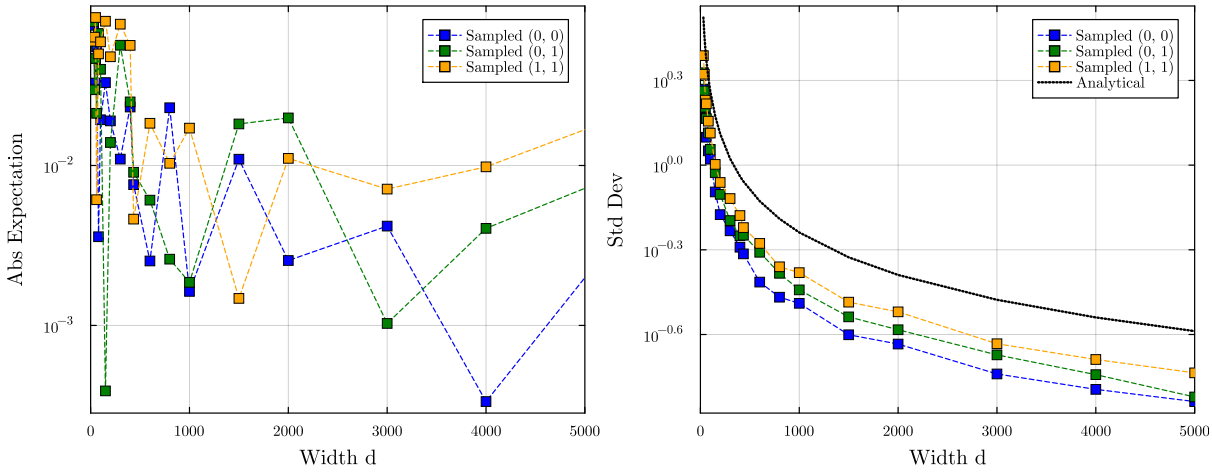


Figure 26: Gaussian Construction: Error Expectation and Std Dev vs Width.

For Rademacher and Gaussian constructions, we see that the absolute expectations are of the order of 10^{-2} and significantly smaller than the expectation of the Bernoulli construction, which matches the theoretical expectation of the symmetric construction. For the standard deviation, we see qualitatively very similar behavior; the analytical estimate significantly overestimates the sampled statistics but captures the qualitative behavior well. Notably, the Rademacher construction has consistently lower standard deviation than the Gaussian construction.

A.4.5 Details for Comparison to Trained Neural Networks

For comparison, we train neural networks to approximate the U-And function and evaluate the error on random sparse inputs. See Section 4.4.2 for plots.

In addition to the previous *CN* case, we here consider a *trained network* (TN) case, where we train neural networks with the same architecture as the constructions.

We train the networks on random sparse Boolean batches, where each entry has $s = 5$ non-zero entries. The batches are sampled uniformly at random at each step with a batch size of 64. We use

the `AdamW` optimizer with a learning rate of 0.0064 and minimize the mean squared error. To improve training stability, we employ a two-phase training schedule:

(i) **Initial Training**: We train for 5000 steps using a $\mathrm{LeakyReLU}$ activation with slope 0.1.
(ii) **Finetuning**: We switch the activation to ReLU and finetune for another 2000 steps.

The networks are initialized with Gaussian initialization with a standard deviation of 0.01.

One central challenge is the read-off layer size. For large m, computing the full read-off layer of size $\binom{m}{2} \times d$ is computationally expensive. Therefore, we train on small random subsets of the read-off at each step. Specifically, for each batch, we include all 'active' pairs (where the target is 1) and sample an additional 32 random pairs (where the target is likely 0) to provide a balanced gradient signal.

A.5 Closed Form Representations of the Loss

We aim to obtain a well-computable representation of the l_1-loss on the homogeneous polygon configuration manifold that does not rely on numerical integration. Numerical integration introduces problems for the analysis. For instance, when using BFGS to optimize the loss, one obtains different numbers of solutions depending on the precision of the numerical integration. In this section, we derive two forms of representations for the loss: first, a general representation that only eliminates expectations in Appendix A.5.1 and second, a closed form only depending on the two parameters ρ, γ in Appendix A.5.2.

A.5.1 Expectation-Free Representation of the General Toy Model l_1-Loss

Proposition 1.22. (Closed Loss Representation Without Expectations):
Let $\boldsymbol{W}, b$ be a general weight-bias configuration of the toy model and let $Y \sim \mathcal{U}[0,1]$. Define for each feature i the quantities

- $\rho_i := \left\|W^i\right\|_2^2$,
- $\gamma_i := -\frac{b_i}{\rho_i}$ and
- $\gamma_i' := \operatorname{clamp}(\gamma_i, [0,1])$

as well as for each feature pair (i,j) the quantities

- $\rho_{ij} := W^i \cdot W^j$,
- $\gamma_{ij} := -\frac{b_j}{\rho_{ij}}$ and
- $\gamma_{ij}' := \operatorname{clamp}\big(\gamma_{ij}, [0,1]\big)$.

Then, the loss term l_1 from (1.2) can be expressed as

$$l_1 = \sum_i I_i B_i + \sum_i \sum_{j \neq i} I_j \mathrm{If}_{ij},$$

where the benefit term B_i is given by

$$B_i = \begin{cases} \frac{1}{3} - \operatorname{ReLU}(b_i) + \operatorname{ReLU}(b_i)^2 & \text{if } \rho_i = 0, \\ \frac{1}{3}(\gamma_i')^3 + \frac{1}{3}(1-\rho_i)^2\left(1-(\gamma_i')^3\right) - (1-\rho_i)b_i\left(1-(\gamma_i')^2\right) + b_i^2(1-\gamma_i') & \text{if } \rho_i > 0. \end{cases}$$

The interference term If_{ij} is given by

$$\mathrm{If}_{ij} = \begin{cases} 0 & \text{if } \ \rho_{ij} + b_j \leq 0 \text{ and } \ b_j \leq 0, \\ \frac{\rho_{ij}^2}{3}\left(1-\left(\gamma_{ij}'\right)^3\right) + 2\rho_{ij}b_j\left(1-\left(\gamma_{ij}'\right)^2\right) + b_j^2\left(1-\gamma_{ij}'\right) & \text{if } \ \rho_{ij} > 0 \text{ and } \ \rho_{ij} + b_j > 0, \\ \frac{\rho_{ij}^2}{3}\left(\gamma_{ij}'\right)^3 + 2\rho_{ij}b_j\left(\gamma_{ij}'\right)^2 + b_j^2\gamma_{ij}' & \text{if } \ \rho_{ij} < 0 \text{ and } \ b_j > 0. \end{cases}$$

Proof.
We start from the general loss representation from (1.2):

$$\begin{aligned} l_1 &= \sum_i I_i \, \mathbb{E}\left[\left(Y - \operatorname{ReLU}\left(\left\|W^i\right\|_2^2 Y + b_i\right)\right)^2\right] + \sum_i \sum_{j \neq i} I_j \, \mathbb{E}\left[\operatorname{ReLU}\left(W^j \cdot W^i Y + b_j\right)^2\right] \\ &=: \sum_i I_i B_i + \sum_i \sum_{j \neq i} I_j \mathrm{If}_{ij}, \end{aligned}$$

where we denote the benefit terms by B_i and the interference term by If_{ij}. We first handle the benefit terms.

First, if $\rho_i = 0$, then the benefit term reduces to

$$B_i = \mathbb{E}\big[(Y - \mathrm{ReLU}(0Y + b_i))^2\big] = \mathbb{E}\big[(Y - \mathrm{ReLU}(b_i))^2\big] = \frac{1}{3} - \mathrm{ReLU}(b_i) + \mathrm{ReLU}(b_i)^2.$$

If $\rho_i > 0$, we have that $\gamma_i > 0$ is well-defined and

$$\begin{aligned}
B_i &= \mathbb{E}\left[\left(Y - \mathrm{ReLU}\left(\left\|W^i\right\|_2^2 Y + b_i\right)\right)^2\right] \\
&= \mathbb{E}\big[(Y - \mathrm{ReLU}(\rho_i Y + b_i))^2\big] \\
&= \mathbb{E}\big[Y^2 1_{Y \le \gamma_i}\big] + \mathbb{E}\big[((1-\rho_i)Y - b_i)^2 1_{Y > \gamma_i}\big] \\
&= \mathbb{E}\big[Y^2 1_{Y \le \gamma_i}\big] + \mathbb{E}\big[\big((1-\rho_i)^2 Y^2 - 2(1-\rho_i) b_i Y + b_i^2\big) 1_{Y > \gamma_i}\big] \\
&= \frac{1}{3}\gamma_i'^3 + \frac{1}{3}(1-\rho_i)^2 [y^3]_{\gamma_i'}^1 - (1-\rho_i) b_i [y^2]_{\gamma_i'}^1 + b_i^2 [y]_{\gamma_i'}^1. \\
&= \frac{1}{3}\gamma_i'^3 + \frac{1}{3}(1-\rho_i)^2 (1 - \gamma_i'^3) - (1-\rho_i) b_i (1 - \gamma_i'^2) + b_i^2 (1 - \gamma_i').
\end{aligned}$$

For the interference term, we define

- $\rho_{ij} \coloneqq W^i \cdot W^j$
- $\gamma_{ij} \coloneqq -\frac{b_j}{\rho_{ij}}$ and
- $\gamma'_{ij} \coloneqq \mathrm{clamp}\big(\gamma_{ij}, [0, 1]\big)$.

Now, we have three cases:

- If $\rho_{ij} + b_j \le 0$ and $b_j \le 0$, then the ReLU is always inactive and we have $\mathrm{If}_{ij} = 0$.
- For sufficiently strong positive interference, i.e., $\rho_{ij} + b_j > 0$ and $\rho_{ij} > 0$, we have

$$\begin{aligned}
\mathrm{If}_{ij} &= \mathbb{E}\Big[\mathrm{ReLU}\big(W^j \cdot W^i Y + b_j\big)^2\Big] \\
&= \mathbb{E}\Big[\big(\rho_{ij} Y + b_j\big)^2 1_{Y \ge \gamma_{ij}}\Big] \\
&= \mathbb{E}\big[\rho_{ij}^2 Y^2 + 2\rho_{ij} b_j Y + b_j^2 1_{Y \ge \gamma_{ij}}\big] \\
&= \frac{\rho_{ij}^2}{3} [y^3]_{\gamma'_{ij}}^1 + 2\rho_{ij} b_j [y^2]_{\gamma'_{ij}}^1 + b_j^2 [y]_{\gamma'_{ij}}^1 \\
&= \frac{\rho_{ij}^2}{3}\big(1 - \gamma_{ij}'^3\big) + 2\rho_{ij} b_j (1 - \gamma_{ij}'^2) + b_j^2 \big(1 - \gamma'_{ij}\big).
\end{aligned}$$

- For negative interference, i.e., $\rho_{ij} < 0$ and a positive bias $b_j > 0$, we have

$$\begin{aligned}
\mathrm{If}_{ij} &= \mathbb{E}\Big[\mathrm{ReLU}\big(\rho_{ij} Y + b_j\big)^2\Big] \\
&= \mathbb{E}\Big[\big(\rho_{ij} Y + b_j\big)^2 1_{Y \le \gamma_{ij}}\Big] \\
&= \mathbb{E}\big[\rho_{ij}^2 Y^2 + 2\rho_{ij} b_j Y + b_j^2 1_{Y \le \gamma_{ij}}\big] \\
&= \frac{\rho_{ij}^2}{3} [y^3]_0^{\gamma'_{ij}} + 2\rho_{ij} b_j [y^2]_0^{\gamma'_{ij}} + b_j^2 [y]_0^{\gamma'_{ij}} \\
&= \frac{\rho_{ij}^2}{3}\gamma_{ij}'^3 + 2\rho_{ij} b_j \gamma_{ij}'^2 + b_j^2 \gamma'_{ij}.
\end{aligned}$$

  (Notably, this case is not advantageous for the model, as reducing the bias would always reduce the interference and thus the loss. Thus, this case likely disappears after a few optimization steps.)

□

A.5.2 Closed Form of the Loss for Homogeneous Polygon Configurations with Continuous Dependence

Here, we derive a closed form of the loss, which allows continuous dependence on the number of features m via the cosines of the angles between feature vectors, given a fixed number of interfering neighbors $2k$. This allows us to analyze the loss landscape in more detail.

Lemma 1.23. (Analytical Loss on the Homogeneous Polygon Manifold):
Let a homogeneous polygon configuration be parameterized by $\beta = b_1 = \cdots = b_m$ and $\rho = r^2 = \left\|W^1\right\|_2^2 = \cdots = \left\|W^m\right\|_2^2$, with $\Delta\varphi := \frac{2\pi}{m}$. Define
- $\gamma := -\frac{\beta}{\rho}$,
- $c_{\Delta \mathrm{i}} := \cos(\Delta \mathrm{i} \Delta\varphi)$,
- $\gamma'_{\Delta \mathrm{i}} := \operatorname{clamp}\left(\frac{\gamma}{c_{\Delta \mathrm{i}}}, [0,1]\right)$ and
- $Y \sim \mathcal{U}[0,1]$.

Let $0 \leq \gamma \leq 1$.

Then, the loss on the homogeneous polygon configuration manifold can be expressed as

$$l_1(\rho, \gamma) = m\left(B(\rho, \gamma) + \sum_{\Delta \mathrm{i}=1}^{m-1} I_{\Delta \mathrm{i}}(\rho, \gamma)\right).$$

where the benefit term $B(\rho, \gamma)$ is given by

$$\begin{aligned} B(\rho, \gamma) &= \mathbb{E}\left[Y^2 1_{Y \leq \gamma}\right] + \mathbb{E}\left[((1-\rho)Y + \rho\gamma)^2 1_{Y > \gamma}\right] \\ &= \frac{1}{3}\gamma^3 + \frac{1}{3}(1-\rho)^2(1-\gamma^3) + (1-\rho)\rho\gamma(1-\gamma^2) + (\rho\gamma)^2(1-\gamma). \end{aligned}$$

The interference term $I_{\Delta \mathrm{i}}(\rho, \gamma)$ is given by

$$\begin{aligned} I_{\Delta \mathrm{i}}(\rho, \gamma) &= \mathbb{E}\left[\mathrm{ReLU}(\rho \cos(\Delta \mathrm{i} \Delta\varphi) Y + \beta)^2\right] \\ &= \begin{cases} \frac{\rho^2}{3c_{\Delta \mathrm{i}}}(c_{\Delta \mathrm{i}} - \gamma)^3 - \frac{\rho^2}{3c_{\Delta \mathrm{i}}}(c_{\Delta \mathrm{i}}\gamma'_{\Delta \mathrm{i}} - \gamma)^3 & \text{if } c_{\Delta \mathrm{i}} > 0 \\ 0 & \text{if } c_{\Delta \mathrm{i}} \leq 0 \end{cases}. \end{aligned}$$

Proof.
We start from the loss in (1.2). We first rewrite the l_1 loss in terms of homogeneous parameters $\beta = b_1 = \cdots = b_m$ and $\rho = r^2 = \left\|W^1\right\|_2^2 = \cdots = \left\|W^m\right\|_2^2$, setting all importances to 1. We define $\Delta\varphi := \frac{2\pi}{m}$ and let $\Delta \mathrm{i}$ denote the index difference.

$$\begin{aligned} l_1 &= \sum_i \underbrace{\mathbb{E}\left[(Y - \mathrm{ReLU}(\rho Y + \beta))^2\right]}_{=:B(\text{Benefit})} + \sum_i \sum_{j \neq i} \underbrace{\mathbb{E}\left[\mathrm{ReLU}(\rho \cos((j-i)\Delta\varphi) Y + \beta)^2\right]}_{=:I_{\Delta \mathrm{i}}(\text{Interference})}. \\ &= mB + m \sum_{\Delta \mathrm{i}=1}^{m-1} I_{\Delta \mathrm{i}} \\ &= m\left(B + \sum_{\Delta \mathrm{i}=1}^{m-1} I_{\Delta \mathrm{i}}\right) \end{aligned}$$

We now want to handle the expectations. We have $Y \sim \mathcal{U}[0,1]$. The expectations split into two parts, one where the ReLU is active, and one where it is not. For example, looking at the first term, we have

$$\mathbb{E}[(Y - \mathrm{ReLU}(\rho Y + \beta))^2] = \mathbb{E}\big[Y^2 1_{\rho Y+\beta \le 0}\big] + \mathbb{E}\big[(Y - (\rho Y + \beta))^2 1_{\rho Y+\beta > 0}\big].$$

We have $\rho Y + \beta \le 0 \Leftrightarrow Y \le -\frac{\beta}{\rho}$. The bias-radius ratio $\gamma := -\frac{\beta}{\rho}$ is a key factor in our investigation. We reformulate the benefit term in terms of $\gamma = -\frac{\beta}{\rho}$ instead of β:

$$\begin{aligned} B &= \mathbb{E}[(Y - \mathrm{ReLU}(\rho Y + \beta))^2] \\ &= \mathbb{E}\big[Y^2 1_{Y \le \gamma}\big] + \mathbb{E}\big[(Y - (\rho Y + \beta))^2 1_{Y > \gamma}\big] \\ &= \mathbb{E}\big[Y^2 1_{Y \le \gamma}\big] + \mathbb{E}\big[(Y - (\rho Y - \rho\gamma))^2 1_{Y > \gamma}\big] \\ &= \mathbb{E}\big[Y^2 1_{Y \le \gamma}\big] + \mathbb{E}\big[((1-\rho) Y + \rho\gamma)^2 1_{Y > \gamma}\big] \end{aligned}$$

We can now expand the expectations. For $Y \sim \mathcal{U}[0,1]$, we have

$$\mathbb{E}\big[(\alpha Y + c)^2 1_{Y>\gamma}\big] = \frac{1}{3\alpha}(\alpha y + c)^3 \;|_{\gamma}^{1}$$

for constants c, α. Thus,

$$\begin{aligned} B(\rho,\gamma) &= \mathbb{E}\big[Y^2 1_{Y \le \gamma}\big] + \mathbb{E}\big[((1-\rho) Y + \rho\gamma)^2 1_{Y > \gamma}\big] \\ &= \mathbb{E}\big[Y^2 1_{Y \le \gamma}\big] + \mathbb{E}\big[((1-\rho)^2 Y^2 + 2(1-\rho)\rho\gamma Y + (\rho\gamma)^2) 1_{Y > \gamma}\big] \\ &= \frac{1}{3}\gamma^3 + \frac{1}{3}(1-\rho)^2(1^3 - \gamma^3) + (1-\rho)\rho\gamma(1^2 - \gamma^2) + (\rho\gamma)^2(1-\gamma) \\ &= \frac{1}{3}\gamma^3 + \frac{1}{3}(1-\rho)^2(1 - \gamma^3) + (1-\rho)\rho\gamma(1 - \gamma^2) + (\rho\gamma)^2(1-\gamma) \end{aligned}$$

Consider the interference term $I_{\Delta\mathrm{i}}(\rho,\gamma)$, which is given by

$$\begin{aligned} I_{\Delta\mathrm{i}}(\rho,\gamma) &= \mathbb{E}[\mathrm{ReLU}(\rho\cos(\Delta\mathrm{i}\Delta\varphi) Y + \beta)^2] \\ &= \mathbb{E}\big[(\rho c_{\Delta\mathrm{i}} Y + \beta)^2 1_{\rho c_{\Delta\mathrm{i}} Y + \beta > 0}\big]. \end{aligned}$$

We distinguish two cases for $c_{\Delta\mathrm{i}}$:

- If $c_{\Delta\mathrm{i}} \le 0$, since $\gamma \ge 0$ (implying $\beta \le 0$) and $Y \ge 0$, we have $\rho c_{\Delta\mathrm{i}} Y + \beta \le 0$ for all Y. Thus, the interference term is zero.
- If $c_{\Delta\mathrm{i}} > 0$, we have $\rho c_{\Delta\mathrm{i}} Y + \beta > 0 \Leftrightarrow Y > -\frac{\beta}{\rho c_{\Delta\mathrm{i}}} = \frac{\gamma}{c_{\Delta\mathrm{i}}}$. Thus,

$$\begin{aligned} I_{\Delta\mathrm{i}}(\rho,\gamma) &= \mathbb{E}\Big[(\rho c_{\Delta\mathrm{i}} Y - \rho\gamma)^2 1_{Y \ge \frac{\gamma}{c_{\Delta\mathrm{i}}}}\Big] \\ &= \mathbb{E}\Big[\rho^2 (c_{\Delta\mathrm{i}} Y - \gamma)^2 1_{Y \ge \frac{\gamma}{c_{\Delta\mathrm{i}}}}\Big] \\ &= \frac{\rho^2}{3c_{\Delta\mathrm{i}}}(c_{\Delta\mathrm{i}} y - \gamma)^3 \;|_{\gamma'_{\Delta\mathrm{i}}}^{1} \\ &= \frac{\rho^2}{3c_{\Delta\mathrm{i}}}(c_{\Delta\mathrm{i}} - \gamma)^3 - \frac{\rho^2}{3c_{\Delta\mathrm{i}}}(c_{\Delta\mathrm{i}}\gamma'_{\Delta\mathrm{i}} - \gamma)^3, \end{aligned}$$

where $\gamma'_{\Delta\mathrm{i}} := \mathrm{clamp}\big(\frac{\gamma}{c_{\Delta\mathrm{i}}}, [0,1]\big)$.

□

A.6 AI Usage Declaration

- Tool Usage
  - Github Copilot Inline Editor: Used for typing assistance and code suggestions in Python, Julia, Typst files.
  - Github Copilot Agent:
    - English grammar and style improvements.
    - Rephrasing sentences for clarity and conciseness.
    - Editing files for consistency and syntax correction.
  - ChatGPT, Google, including Gemini:
    - Search assistance for relevant literature and references.
- All AI-generated content was reviewed and edited by the author to ensure accuracy and coherence with the thesis content.
- The use of AI tools was limited to language and code assistance; all except declared intellectual contributions and original ideas are those of the author.
  - the proof of Lemma 1.3 was based on suggestions from Gemini. The final formulation and proof are the author's own work. To the best of our knowledge, the statement is a standard result in Boolean function analysis; it does not play a central role in the analysis but is included for completeness.

Bibliography

[1] N. Elhage *et al.*, "Toy models of superposition," *arXiv preprint*, 2022, doi: 10.48550/arXiv.2209.10652.

[2] K. Hänni, J. Mendel, D. Vaintrob, and L. Chan, "Mathematical Models of Computation in Superposition," in *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. doi: 10.48550/arXiv.2408.05451.

[3] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom in: An introduction to circuits," *Distill*, vol. 5, no. 3, pp. e24–1, 2020, doi: 10.23915/distill.00024.001.

[4] N. Cammarata, G. Goh, S. Carter, L. Schubert, M. Petrov, and C. Olah, "Curve Detectors," *Distill*, 2020, doi: 10.23915/distill.00024.003.

[5] A. Templeton *et al.*, "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet," *Transformer Circuits Thread*, 2024, [Online]. Available: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

[6] G. Goh *et al.*, "Multimodal Neurons in Artificial Neural Networks," *Distill*, 2021, doi: 10.23915/distill.00030.

[7] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 1877–1901. doi: 10.48550/arXiv.2005.14165.

[8] T. Henighan *et al.*, "Superposition, Memorization, and Double Descent," *Transformer Circuits Thread*, 2023, [Online]. Available: https://transformer-circuits.pub/2023/toy-double-descent/index.html

[9] N. Elhage *et al.*, "Softmax Linear Units," *Transformer Circuits Thread*, 2022, [Online]. Available: https://transformer-circuits.pub/2022/solu/index.html

[10] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," *arXiv preprint*, 2023, doi: 10.48550/arXiv.2309.08600.

[11] D. Braun, L. Bushnaq, S. Heimersheim, J. Mendel, and L. Sharkey, "Interpretability in Parameter Space: Minimizing Mechanistic Description Length with Attribution-based Parameter Decomposition," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2501.14926.

[12] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing Higher-Layer Features of a Deep Network," Technical Report 1341, June 2009. doi: 10.48550/arXiv.1301.3605.

[13] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *European Conference on Computer Vision (ECCV)*, in Lecture Notes in Computer Science, vol. 8689. Springer, 2014, pp. 818–833. doi: 10.1007/978-3-319-10590-1_53.

[14] C. Olah *et al.*, "The Building Blocks of Interpretability," *Distill*, 2018, doi: 10.23915/distill.00010.

[15] J. Jo and Y. Bengio, "Measuring the tendency of CNNs to Learn Surface Statistical Regularities," *arXiv preprint*, 2017, doi: 10.48550/arXiv.1711.11561.

[16] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2019, pp. 1909–1920. doi: 10.48550/arXiv.1905.02175.

[17] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint*, 2018, doi: 10.48550/arXiv.1811.12231.

[18] S. Black *et al.*, "Interpreting Neural Networks through the Polytope Lens." Accessed: Nov. 15, 2025. [Online]. Available: https://www.alignmentforum.org/posts/eDicGjD9yte6FLSie/interpreting-neural-networks-through-the-polytope-lens

[19] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt, "Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small," *arXiv preprint*, 2022, doi: 10.48550/arXiv.2211.00593.

[20] S. J. Thorpe, "Local vs. Distributed Coding," *Intellectica*, vol. 8, no. 2, pp. 3–40, 1989, doi: 10.3406/intel.1989.873.

[21] G. E. Hinton, "Distributed representations," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press, Cambridge, MA, pp. 77–109, 1986.

[22] N. Elhage *et al.*, "A Mathematical Framework for Transformer Circuits," *Transformer Circuits Thread*, 2021, [Online]. Available: https://transformer-circuits.pub/2021/framework/index.html

[23] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 2013, pp. 746–751. [Online]. Available: https://aclanthology.org/N13-1090/

[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *International Conference on Learning Representations (ICLR)*, 2016. doi: 10.48550/arXiv.1511.06434.

[25] N. Nanda, A. Lee, and M. Wattenberg, "Emergent Linear Representations in World Models of Self-Supervised Sequence Models," in *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, Y. Belinkov, S. Hao, J. Jumelet, N. Kim, A. McCarthy, and H. Mohebbi, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 16–30. doi: 10.18653/v1/2023.blackboxnlp-1.2.

[26] K. Park, Y. J. Choe, and V. Veitch, "The linear representation hypothesis and the geometry of large language models," *arXiv preprint*, 2023, doi: 10.48550/arXiv.2311.03658.

[27] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski, "Linear algebraic structure of word senses, with applications to polysemy," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 483–495, 2018, doi: 10.1162/tacl_a_00034.

[28] M. Adler and N. Shavit, "On the Complexity of Neural Computation in Superposition," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2409.15318.

[29] L. Bushnaq and J. Mendel, "Circuits in Superposition: Compressing many small neural networks into one." Accessed: Nov. 08, 2025. [Online]. Available: https://www.alignmentforum.org/posts/roE7SHjFWEoMcGZKd/circuits-in-superposition-compressing-many-small-neural

[30] L. Linsefors and L. Bushnaq, "Circuits in Superposition 2: Now with Less Wrong Math." Accessed: Nov. 08, 2025. [Online]. Available: https://www.alignmentforum.org/posts/FWkZYQceEzL84tNej/circuits-in-superposition-2-now-with-less-wrong-math

[31] L. Sharkey, D. Braun, and B. Millidge, "Taking features out of superposition with sparse autoencoders." Accessed: Oct. 28, 2025. [Online]. Available: https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-outof-superposition

[32] T. Bricken *et al.*, "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning," *Transformer Circuits Thread*, 2023, [Online]. Available: https://transformer-circuits.pub/2023/monosemantic-features/index.html

[33] D. Chanin, H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "A is for Absorption: Studying Feature Splitting and Absorption in Sparse Autoencoders," *arXiv preprint*, 2024, doi: 10.48550/arXiv.2409.14507.

[34] J. Engels, E. J. Michaud, I. Liao, W. Gurnee, and M. Tegmark, "Not All Language Model Features Are One-Dimensionally Linear," in *International Conference on Learning Representations (ICLR)*, 2025. doi: 10.48550/arXiv.2405.14860.

[35] L. Bushnaq, D. Braun, and L. Sharkey, "Stochastic Parameter Decomposition," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2506.20790.

[36] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003, doi: 10.1002/rsa.10073.

[37] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint*, 2013, doi: 10.48550/arXiv.1312.6120.

[38] "Direct sum." Accessed: Oct. 28, 2025. [Online]. Available: https://polytope.miraheze.org/wiki/Direct_sum?oldid=111021

[39] "Pentagonal tegum image." Accessed: Oct. 28, 2025. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/d/de/Pentagonal_tegum.svg

[40] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[41] S. Boucheron, G. Lugosi, and O. Bousquet, "Concentration Inequalities," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 208–240. doi: 10.1007/978-3-540-28650-9_9.

[42] M. Walters, "Concentration of measure techniques and applications," Doctoral dissertation, University of Rochester, 2015. doi: 10.48550/arXiv.1508.05448.

[43] A. Klenke, *Probability Theory: A Comprehensive Course*, 2nd ed. in Universitext. Springer London, 2014. doi: 10.1007/978-1-4471-5361-0.

[44] C. Olah, "Distributed Representations: Composition & Superposition," *Transformer Circuits Thread*, 2023, [Online]. Available: https://transformer-circuits.pub/2023/superposition-composition/index.html

[45] E. Ameisen *et al.*, "Circuit Tracing: Revealing Computational Graphs in Language Models," *Transformer Circuits Thread*, 2025, [Online]. Available: https://transformer-circuits.pub/2025/attribution-graphs/methods.html