Abstract

Recent work in neural network interpretability has posited that hidden activations of some deep models can be viewed as superpositions of sparsely activated "feature vectors." In general, this kind of representation is known as a superposition code. This work presents an information-theoretic account of superposition codes in a setting intended to model applications of SAEs. We show that, when the number k of active features is very small compared to the number N of total possible features, surprisingly simple superposition methods can encode and decode these representations provided that d is a constant factor greater than the Shannon limit. Specifically, when $\ln k / \ln N \leq \epsilon < 1$, it suffices that $C(\eta)d \geq H$, where H is the entropy to be transmitted and $C(\eta)$ is a certain decreasing function of η.. However, the maximum value of d/H varies significantly depending on the decoding method used. For example, under moderate values of η, we find that certain natural decoding methods are limited to transmission rates of less than $1/6$ bits per dimension. On the other hand, we show empirically that a cheap iterative method can achieve around $3/4$ bits per dimension. Our results address a lack of practical references on superposition codes in a very sparse regime and provide a possible information-theoretic explanation for the limited success of SAEs.

1 Introduction

If each neuron in a given neural network coded for a "meaningful" feature of its input, we could hope to reverse-engineer this network's overall behavior on a neuron-by-neuron basis. However, individual neurons of real-world networks often lack clear interpretations. For example, both language models and vision models have been found to learn neurons that correlate simultaneously with apparently unrelated features. (See for example Nguyen et al. (2016), Zhang & Wang (2023) and Olah et al. (2020).)

The difficulty of interpreting a network in terms of its local activity—and in particular, the appearance of so-called "polysemantic neurons"—is not surprising from a connectionist viewpoint. Since at least the 1980s, proponents of neural networks have argued that these systems may naturally use **distributed representations**—coding schemes where individual features are represented by patterns spread over many neurons, and conversely where each neuron carries information on many features. (This term was apparently coined in Rumelhart et al. (1986), Chapter 3.) In contrast, a *local* representation would dedicate each neuron to a single feature. See Thorpe (1989) for a general discussion of local and distributed codes. Figure 1 illustrates a classic example of a coarse code, one kind of distributed representation.

As of now, relatively little is known about how deep neural networks learn to represent information in their hidden layers or to what extent this information can be interpreted. However, should "interpretable features" exist, the connectivist viewpoint makes it natural that they would typically be stored with non-local codes. This is a common assumption in interpretability research today; for example, when Meng et al. (2022)

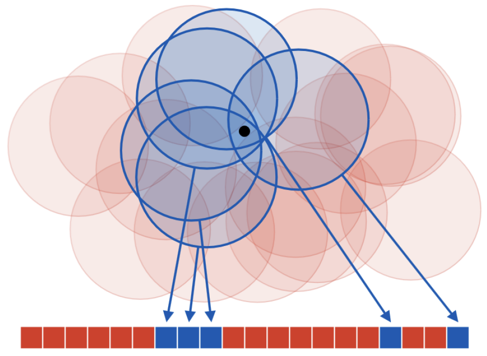

Figure 1: A coarse code representing a point on a plane. Each "neuron," drawn as a red or blue square, encodes whether the point belongs to an associated "receptive field." Although no neuron gives specific information on the position of the point, the overall code determines its position with reasonable accuracy.

intervened on an MLP layer of a language model to "edit" a factual association, both the "subject" and the "fact" were modeled as vectors of neuron activations rather than as individual neurons.

How can we infer latent features learned by a neural network? One simple proposal is to model an activation vector x as a linear projection

$$x = Fy$$

of some high-dimensional and *sparse* vector y of latent features. We refer to the columns of F as codewords and the whole matrix F as a dictionary. Since x is a linear superposition of codewords, we will call it a **superposition code** for y. It is a remarkable fact that, for certain matrices F, the linear projection $x = Fy$ can code uniquely for a sparse vector y even when the dimension of x is (much) smaller than the dimension of y. The task of inferring the sparse vector from its linear projection is known as sparse reconstruction, and the task of inferring the dictionary F from a distribution over x is called dictionary learning. Both of these problems have been studied in the field of compressive sensing, although with different applications in mind. (See Elad (2010) for a review of classic work in the context of signal and image processing.)

Already in 2015, Faruqui et al. (2015) used a dictionary learning method to derive sparse latent codes for word embeddings and argued that these latents were more interpretable than the original embedding dimensions. More recently, a series of works beginning with Yun et al. (2021) have applied dictionary learning to the internal representations of transformer-based language models. Cunningham et al. (2023) suggested the use of **sparse autoencoders** (SAEs) and Templeton et al. (2024); Gao et al. (2024) scaled sparse autoencoders to production-size large language models.

To infer a latent representation $y \in \mathbb{R}^N$ from an activation vector $x \in \mathbb{R}^d$, sparse autoencoders use an estimate like $\hat{y}(x) = \sigma(Gx)$ for some learnable matrix $G\colon \mathbb{R}^{N \times d}$ and some simple non-linear thresholding function σ. Given a latent vector $\hat{y}$, the In Gao et al. (2024), the number of latent dimensions considered was on the order of $N = 10^6$ and latent representations y were assumed to have on the order of $k = 10^2$ non-zero coefficients.

Templeton et al. (2024) showed that latent features learned by SAEs are often highly intepretable, and that intervention on these features allows "steering" language models in predictable ways. However, as reported in Gao et al. (2024), even SAEs with extremely large numbers of latents suffer from an apparently irreducible reconstruction error. According to Sharkey et al. (2025), understanding the limitations of SAEs—and dictionary learning in general—is an important open question in the research program of mechanistic interpretability.

Contributions

Over the course of computation, it is common for programs to use more memory than was required to store their input data. Similarly, it is natural to expect that the entropy of a "simple" (or "interpretable") description for the residual vectors within a language model is not bounded by the entropy of language. One natural question is *how much* information may hypothetically be read off from an activation vector.

In many classical situations, this question is answered by well-known ideas from information theory. Given a "channel" with certain characteristics—for example, a band-limited telephone connection or a binary storage device with a certain failure rate—the amount of information we can effectively transmit is asymptotically characterized by a "channel capacity," measured in bits per unit of channel usage. The goal of this paper is to study the carrying capacity of sparse superposition codes in a regime that is applicable to sparse autoencoders. More specifically, we consider the number of dimensions d required for a d-dimensional superposition code to ... explain more here

Our analysis focuses on a toy example, described in more detail in Section 2. Our main conclusions are the following.

- In a very sparse regime ($\ln k \leq \epsilon \ln N$ for $\epsilon < 1$), sparse superposition codes can be decoded by estimating each latent feature independently, so long

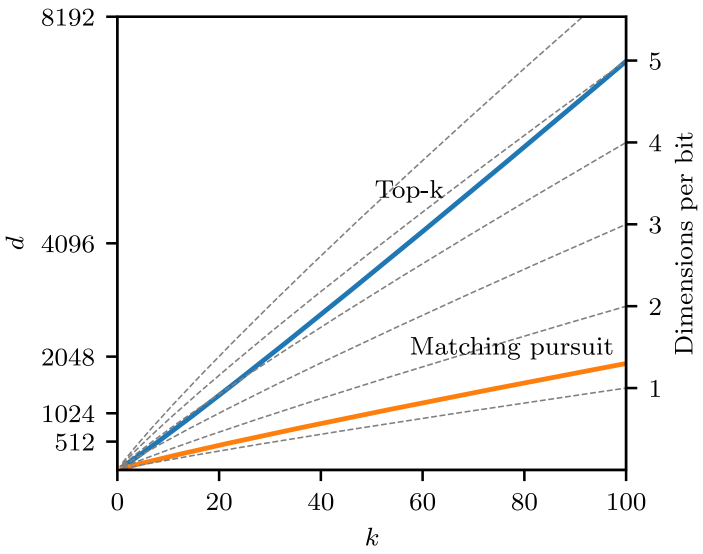

Figure 2: An overview of the minimum codeword dimension d required for two different methods to reliably decode a uniformly chosen k-sparse subset of $\{1,\ldots,2^{20}\}$ from a superposition of codewords from an i.i.d. Rademacher dictionary. The inverse "bitrate" $d/H(k)$, where $H(k) = \log_2 \binom{N}{k} \approx k\log_2(eN/k)$, is indicated by the right axis. Top-k is a method currently used by sparse autoencoders, and matching pursuit is a relatively cheap iterative algorithm.

How "efficient," in terms of bitrate, are the codes used by real neural networks? Of course, it would not make sense for a network to use a code that requires a costly iterative decoding process before it can be used. However, the success of matching pursuit suggests that neural networks may be able to improve over the bitrates of one-step estimates while paying a relatively small computational price. Although our analysis is restricted to a toy scenario, we hope these results inform future work on modeling distributed representations.

Related Work

Recently, various authors have studied the underperformance of SAEs and proposed ways to improve these methods. For example, Rajamanoharan et al. (2024) proposed *gated SAEs* to mitigate "feature shrinkage," and Bussmann et al. (2025) proposed *Matryoska SAEs* to deal with problems related to "feature absorption."

Especially relevant to this work is the proposal of *inference-time optimization* (ITO), which involves replacing the encoder of an SAE with an iterative optimization method at inference time—that is, after the dictionary F has already been learned. For example, Engels et al. (2024) evaluated gradient pursuit as an inference-time optimization method. However, if the restrictive encoders used by SAEs at training time cannot discern read certain attributes of the sparse latent signal, then the corresponding codewords will not appear in the learned dictionary. This may explain the limited improvements attained so far by ITO. We hope the present work helps clarify these questions, especially for readers who may not be familiar with ideas from compressive sensing.

2 Superposition Codes and Matched Filters

We begin by describing the toy scenario to be studied. Given a large number N, consider a map F that "encodes" each subset $y \subseteq [N] = \{1,\ldots,N\}$ by a linear combination

$$x = \sum_{i\in y} f_i \in \mathbb{R}^d,$$

where the vectors $\{f_i \in \mathbb{R}^d : i \in [N]\}$ are chosen in advance and where the dimension d of the encoding is expected to be much smaller than N. We call the vectors f_i codewords for the elements of $[N]$ and call the image Fy a superposition code for the set y. It may be useful to view y as a vector in $\{0,1\}^N$ with coefficients

$$y_i = \begin{cases} 1 : i \in y \\ 0 : \text{otherwise} \end{cases}$$

and view F as a matrix of column vectors $[f_1 \ \dots \ f_N]$, called the dictionary. In this work, we'll model our subset as a random variable Y uniformly distributed over the subsets of some fixed size $k \ll N$.

To decode a superposition code, we define two informal classes of methods. Methods of the first class first estimate the coordinates of Y using N linear functions of the code X, and then perform a single thresholding step to refine these estimates. We call these *one-step methods*. One-step methods can be derived by considering how each coordinate Y_i of the code can be estimated from the data

$$X = Y_i f_i + \overbrace{\sum_{j \neq i} Y_j f_j}^{Z}. \tag{1}$$

If we know the codeword $f_i \in \mathbb{R}^d$ but approximate "noise" $Z \in \mathbb{R}^d$ produced by the other codewords in superposition. In signal processing, the problem of recovering an unobserved variable from a noisy process is known as *filtering*.

If we agree to model Z as Gaussian noise, then the problem of estimating

In a linear system with Gaussian noise, like Equation (1), optimal filtering can be done using a linear function of the measurement data. Specifically, suppose Z has mean zero and non-singular covariance Σ, and define an inner product by $\langle v, w \rangle_\Sigma = x^T \Sigma^{-1} y$. Then the posterior of S conditional on X is determined by the function

$$\lambda(X) = \frac{\langle f, X \rangle_\Sigma}{\|f\|_\Sigma^2},$$

which we will call the **matched filter** for S. If $S \in \{0, 1\}$ is a binary variable, a routine calculations shows that the log odds of the posterior on S is given by

$$\begin{aligned} &\ln \frac{\mathrm{P}(S = 1 | X = x)}{\mathrm{P}(S = 0 | X = x)} \\ &= \rho \left(\lambda(x) - \frac{1}{2} \right) + \ln \frac{\mathrm{P}(S = 1)}{\mathrm{P}(S = 0)}, \end{aligned} \tag{2}$$

where $\rho = \|f\|_\Sigma^2$ is the "signal-to-noise ratio" of the filter λ. See **??** for a review.

for some parameter vectors λ_i. When the number k of non-zero coefficients is assumed beforehand, as it is in our scenario, we can also choose the threshold adaptively so that only k of the $\hat{Y}_i$ are non-zero. This is called top-k decoding. Gao et al. (2024) showed that, in practice, top-k autoencoders perform better than their ReLU variants in practice.

We're interested in understanding how large the dimension d needs to be as a function of (N, k) for a given inference method to recover Y. (We do not study the problem of learning the dictionary.) Since Y is a discrete variable, we will focus on conditions for *exact* recovery. We focus on a regime where Y resembles the very sparse latent representations learned by sparse autoencoders trained on large language models. For example, typical values discussed in Gao et al. (2024) are $N = 2^6$ and $k = 100$.

We now return to our original problem. Let's focus on estimating just one scalar Y_i from the sum

$$X = Y_i f_i + \sum_{j \neq i} Y_j f_j.$$

The "noise term" here is not Gaussian, and the exact Bayesian posterior on Y_i turns out to be intractable in general. However, we can try to estimate Y_i by approximating $\sum_{j \neq i} Y_j f_i$ by a Gaussian vector of the same covariance. The corresponding matched filter for Y_i can be understood as a kind of least squares estimate.

In the following, let us assume that the codewords $f_i \in \mathbb{R}^d$ are unit vectors. (It is natural for all the codewords f_i to have the same magnitude if each coefficient Y_i needs to be encoded with the same precision, as they do in our scenario.) If we assume further that the empirical distribution over codewords f_i is approximately isotropic, then the matched filter for Y_i is approximately

$$\lambda_i(X) = \langle f_i, X \rangle.$$

(If the distribution over codewords is not isotropic, we can first apply a linear transformation to "whiten" the distribution of X.)

A **one-step estimate** is an estimate for Y that relies directly on the matched filters λ_i. From Equation (2), the maximum likelihood estimate for Y_i under our simplified Gaussian model is 1 if

$$\langle f_i, X \rangle \geq \frac{1}{\rho} \ln \frac{\mathrm{P}(Y_i = 1)}{\mathrm{P}(Y_i = 0)} + \frac{1}{2}$$

and 0 otherwise. If we assume the signal-to-noise ratio ρ is very large, the decision boundary becomes approximately 1/2. This leads to the simpler of the two one-step estimates that we will consider.

Definition 1 *Given $X = FY$, the* ***threshold decoding*** *is*

$$\hat{Y}_i = \begin{cases} 1 : \langle f_i, X \rangle \geq 1/2 \\ 0 : \textit{otherwise.} \end{cases}$$

On the other hand, if we know (or guess) the size k of the set Y in advance, the following is a natural way to make use of that information. (In the context of sparse autoencoders, this method was introduced by Makhzani & Frey (2014).)

Definition 2 *Given $X = FY$, the* ***top-k decoding*** *is the set $\hat{Y}$ of k elements whose codewords f_i have largest inner products with X. (Ties are broken arbitrarily.)*

Note that whenever threshold decoding succeeds at recovering Y, top-k decoding succeeds as well. Indeed, top-k decoding can be viewed as a kind of threshold decoding where the threshold is chosen optimally as a function of X.

3 Information Theory Bounds

We begin with an illustrative example. Suppose that the latent variable Y to be encoded is a symbol drawn from an alphabet of size N. What is the minimum dimension d such that each value of Y can be represented by a vector $X \in \mathbb{R}^d$?

When the vector X can be stored exactly, the answer to our question simply depends on the number of representable vectors. If the coefficients of X are 16 bit floating point numbers, then each coefficient can store (nearly) 2^{16} states, and overall we need only about $d_{\min} \approx \frac{1}{16} \log_2 N$ dimensions.

However, we expect that activation vectors within real neural networks are subject to various kinds of interference. For example, an activation vector X may need to represent the symbol Y while also holding information on another latent quantity, or it may be subject to explicitly induced noise like dropout. Characterizing the "capacity" of a vector in the presence of interference is a basic problem in coding theory.

One common model is to consider that X is subject to additive white Gaussian noise. More specifically, we let $Z \in \mathbb{R}^d$ be a vector of independent, centered Gaussians each of variance P, and consider the problem of recovering Y from $X + Z$. The following is then a typical (but remarkable) result of information theory.

Proposition 1 *Let the random variable $Y \in [N]$ be uniformly distributed, and let Z be white Gaussian noise of variance P. Suppose there exist a pair of maps $F\colon [N] \to \mathbb{R}^d$ and $G\colon \mathbb{R}^d \to [N]$ so that*

$$G(F(Y) + Z) = Y$$

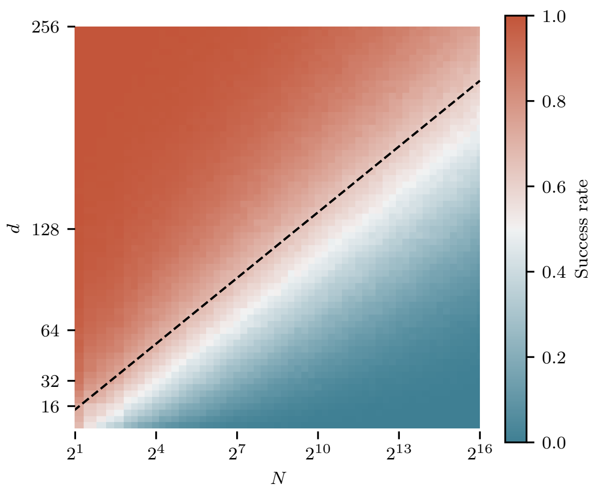

Figure 3: How many dimensions d does the vector $X \in \mathbb{R}^d$ need to have for $X + Z$ to reliably code for an element $Y \in [N]$? This figure shows empirical success of the random codebook strategy when Z is white noise with components of power P.

with probability at least $(1 - p)$, *and suppose the variance of each coordinate of* $F(Y)$ *is bounded by* 1. *Then*

$$d \geq 2\frac{(1-p)\ln N - \ln 2}{\ln\bigl(1 + P^{-1}\bigr)}.$$

In particular, if we fix P and let the number N of symbols grow to infinity, the required dimension d grows logarithmically in N. More specifically, we can say the following.

Corollary 1 *Fix* $P > 0$, $\epsilon > 0$, *and suppose*

$$d \leq \left(\frac{2}{\ln(1 + P^{-1})} - \epsilon\right)\ln N.$$

Then, when the random variables (Y, Z) *and the maps* (F, G) *are in the conditions of Proposition 1, the maximum value of* $\mathrm{P}(G(F(Y) + Z) = Y)$ *converges to* 0 *for sufficiently large* N.

Since $\ln(1 + P^{-1})^{-1} \approx P$ for large P, this means that roughly $d \geq 2P \ln N$ dimensions are needed to reliably code for Y.

Now, one very simple way to construct maps F and G is to choose the codewords $F(i)$ randomly from some distribution on the unit sphere in $\mathbb{R}^d$ and let $G(v)$ return the element j so that $\langle v, F(j)\rangle$ is maximized. For brevity, we refer to (F, G) This strategy can be motivated by the remarkable fact that random codebooks of size $\Omega(e^d)$ of d-dimensional codewords can have bounded interference, in the sense of the following result.

Proposition 2 *Let* (F, G) *be constructed according to the random codebook strategy, let* $\epsilon > 0$, *and let*

$$d \geq (2 + \epsilon)P \ln N.$$

Then, when the random variable (Y, Z) *are in the conditions of Proposition 1,* $\mathrm{P}(G(F(Y)+Z) = Y)$ *converges to* 1 *for large* N.

Altogether, this shows that transmitting

Our discussion so far parallels many Cover & Thomas (2006). In the remainder of this work, we consider

It will also be useful to know the upper bound

$$H = \ln\binom{N}{k} \leq k\ln(eN/k) = k\ln N - k\ln k + k$$

on the entropy of a random k-element subset of $[N]$, which turns out to be a very good approximation when $k \ll N$. For example, when $N = 2^{20}$ and $k = 2^8$, the approximation

$$\ln \binom{2^{20}}{2^8} \approx 2^8 \ln(2^{20}e/2^8) = 128(1 + 12\ln 2)$$

holds with a relative error of only about 0.3%. (See **??** for a discussion of this estimate.)

References

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning Multi-Level Features with Matryoshka Sparse Autoencoders, March 2025. URL `http://arxiv.org/abs/2503.17547`. arXiv:2503.17547 [cs].

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition*. Wiley-Interscience, Hoboken, N.J, 2006. ISBN 978-0-471-24195-9.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models, October 2023. URL `http://arxiv.org/abs/2309.08600`. arXiv:2309.08600.

M. Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer, New York, 2010. ISBN 978-1-4419-7010-7 978-1-4419-7011-4. OCLC: ocn646114450.

Joshua Engels, Logan Riggs, and Max Tegmark. Decomposing The Dark Matter of Sparse Autoencoders, October 2024. URL `http://arxiv.org/abs/2410.14670`. arXiv:2410.14670.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. Sparse Overcomplete Word Vector Representations. In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1491–1500, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1144. URL `https://aclanthology.org/P15-1144/`.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, June 2024. URL `http://arxiv.org/abs/2406.04093`. arXiv:2406.04093 [cs].

Alireza Makhzani and Brendan Frey. k-Sparse Autoencoders, March 2014. URL `http://arxiv.org/abs/1312.5663`. arXiv:1312.5663.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks, May 2016. URL `http://arxiv.org/abs/1602.03616`. arXiv:1602.03616 [cs].

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom In: An Introduction to Circuits. *Distill*, 5(3):10.23915/distill.00024.001, March 2020. ISSN 2476-0757. doi: 10.23915/distill.00024.001. URL `https://distill.pub/2020/circuits/zoom-in`.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving Dictionary Learning with Gated Sparse Autoencoders, April 2024. URL `http://arxiv.org/abs/2404.16014`. arXiv:2404.16014.

David E. Rumelhart, James L. McClelland, and AU. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. The MIT Press, 1986. ISBN 978-0-262-29140-8. doi: 10.7551/mitpress/5236.001.0001.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, Stephen Casper, Max Tegmark, William Saunders, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open Problems in Mechanistic Interpretability, January 2025. URL `http://arxiv.org/abs/2501.16496`. arXiv:2501.16496 [cs].

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. *Transformer Circuits Thread*, May 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

Simon Thorpe. Local vs. Distributed Coding. *Intellectica*, 8(2):3–40, 1989. doi: 10.3406/intel.1989.873. URL `https://www.persee.fr/doc/intel_0769-4113_1989_num_8_2_873`. Publisher: Persée - Portail des revues scientifiques en SHS.

Zeyu Yun, Yubei Chen, Bruno Olshausen, and Yann LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In Eneko Agirre, Marianna Apidianaki, and Ivan Vulić (eds.), *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 1–10, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.deelio-1.1. URL `https://aclanthology.org/2021.deelio-1.1/`.

Changwan Zhang and Yue Wang. A sample survey study of poly-semantic neurons in deep CNNs. In *International Conference on Computer Graphics, Artificial Intelligence, and Data Processing (ICCAID 2022)*, volume 12604, pp. 849–855. SPIE, May 2023. doi: 10.1117/12.2674650.