Relational Composition in Neural Networks: A Survey and Call to Action

Martin Wattenberg [1] **Fernanda B. Viégas** [1]

Abstract

Many neural nets appear to represent data as linear combinations of "feature vectors." Algorithms for discovering these vectors have seen impressive recent success. However, we argue that this success is incomplete without an understanding of **relational composition**: how (or whether) neural nets combine feature vectors to represent more complicated relationships. To facilitate research in this area, this paper offers a guided tour of various relational mechanisms that have been proposed, along with preliminary analysis of how such mechanisms might affect the search for interpretable features. We end with a series of promising areas for empirical research, which may help determine how neural networks represent structured data.

1. Introduction

The *linear representation hypothesis* asserts that neural nets encode information as a sum of "feature vectors" (Mikolov et al., 2013b; Arora et al., 2016; Olah et al., 2020). That is, a layer of a network represents a set of features[2] as a weighted sum of associated vectors. An increasing body of evidence supports this idea, and one recent development is a set of algorithms for automatically finding feature vectors at scale (e.g., (Bricken et al., 2023; Cunningham et al., 2023; Rajamanoharan et al., 2024; Templeton et al., 2024) among others). These new techniques raise the possibility that we can understand and even control neural networks using a catalog of interpretable feature vectors.

The very simplicity of the linear representation hypothesis, however, suggests that there may be more to the story. After all, most computer programs contain data structures far more intricate than mere sets or bags, reflecting the complexity of the world. And programs that do treat memory as just a set of bits or bytes, such as assembly language, are notoriously hard to read and interpret. It's natural to ask, could there be structure in neural representations beyond this "bag of features" model? Indeed, this basic question has attracted notice since the founding of the field (Rosenblatt, 1961; Fodor & Pylyshyn, 1988). A wide array of mechanisms have been proposed for relating and composing vector representations—what we refer to here as **relational composition**.

This paper includes a summary of some of these mechanisms, with a specific focus on ideas that seem relevant to contemporary interpretability work[3]. A central goal is to provide a gateway to a set of work that is spread over multiple decades and disciplines. We also analyze how these mechanisms might interact in subtle ways with feature identification algorithms, potentially causing problems for techniques that "steer" behavior by intervening during inference. In particular, we describe two potential problems that might arise in applying non-compositional methods to compositional representations:

- **Feature multiplicity**. This is a kind of false positive, where combinatorial mechanisms might create multiple distinct "echo" feature vectors that correspond to the same concept. Furthermore, despite representing the same concept, different echo vectors potentially could have different effects on the system's output—leading to what we term a predict/control discrepancy. As a result, using feature vectors to understand and control behavior may be challenging.

- **Dark matter**. This possibility represents a false negative. Some combinatorial mechanisms might systematically hide important features or combinations of features. Moreover, by looking for semantically meaningful vectors, we might miss important relational data. For example, binding mechanisms that involve so-called "ID" vectors (Feng & Steinhardt, 2023) may

[1]Harvard University and Google. Work done at Harvard. Correspondence to: Martin Wattenberg <wattenberg@seas.harvard.edu>, Fernanda Viégas <fernanda@seas.harvard.edu>.

Workshop on Mechanistic Interpretability, Vienna, Austria. Copyright 2024 by the author(s).

[2]See Appendix for a definition of the word "feature."

[3]See (Schlegel et al., 2022) for a more comprehensive historical review of vector compositionality, and (Feldman, 2013) for a general survey of the "binding problem" in biological neural nets.

encode information not by value, but by reference, leading to "features" with no interpretation in terms of system inputs, processing, or outputs.

We're certainly not the first to point out the importance of compositionality in this context. Similar issues are raised in (Bricken et al., 2023) and (Olah, 2024), for example. Our argument here is that the rich set of existing ideas for compositional representations may hold important clues to understanding recent advances in mechanistic interpretability. Even if they don't capture the full workings of neural nets, these mechanisms might be useful human-interpretable approximations. Examining the literature in this light may help us understand the limitations and results of a linear analysis of neural net representations. Given the potential payoff for new results, our aim in this paper is to accelerate research in this area.

2. Definitions, notation, and background

For simplicity, we'll work in an idealized setting, meant to be the bare minimum to explore ideas about composing representations. The basic unit of representation in this setting is a vector $x \in \mathbb{R}^n$. We'll assume that this vector encodes a set of "features" using vector addition:

$$x = \sum_{i=1}^{m} a_i v_i$$

where each v_i is a unit vector representing an individual feature, and the coefficient a_i represents the weight—or sometimes just presence or absence—of that feature in the representation. (See appendix for more on the term "feature.") Note that we may have $m > n$, that is, more features than dimensions; this is called "superposition" in the literature ((Elhage et al., 2022) discusses some of the history of this idea.) In this setting, one typically assumes that features are "sparse", meaning most values of a_i are zero, and that the v_i are nearly mutually orthogonal. These assumptions make it possible to approximately reconstruct feature coefficients via inner products, with

$$a_i \approx \langle x, v_i \rangle$$

This mechanism is good way to represent unordered *sets* of features. But how might we compose these vectors to express structures more complicated than sets? One natural way to combine two feature vectors x and y is simply to add them up. The sum $x + y$ represents something like the union of the features each individual vector encodes. From a geometric point of view, the sum is a vector that is similar to both x and y. This has obvious utility for representations: for instance, one could describe a scene with a duck and a capybara by adding a "duck vector" to a "capybara vector". There's an essential limitation to summation, however, which is that there is no sense of order or other structure. A scene where a duck is perched on a capybara and one where the capybara is perched on the duck would have the same representation. Analyzing this issue leads to deep waters. Indeed, Fodor and Pylyshyn famously argued that neural nets are intrinsically limited in their ability to express more complex, directed relations (Fodor & Pylyshyn, 1988). There have been many, many responses to that argument. One particular type of counterargument is to suggest a mechanism for composing or binding elements in a way to preserves relationships.

Most of these relational composition mechanisms fall into three broad classes, depending on whether they're based on binding vectors with matrices, tensors, or references across tokens. This division doesn't capture the full range of possibilities, but it's enough for our purposes. We now discuss some of these mechanisms, and their potential implications for finding and using features.

3. Additive matrix binding

One general class of composition mechanisms is based on matrix transformations. Suppose $x, y \in \mathbb{R}^n$ are two vectors. How might we represent the ordered pair (x, y) using a single vector $r \in \mathbb{R}^n$? As discussed, we can't just use the sum $x + y$ since this would destroy any sense of order: the pairs (x, y) and (y, x) would have the same representation.

One method that does preserve order is to fix two distinct $n \times n$ matrices A and B and define the representation of (x, y) by

$$r = Ax + By$$

It's clear that (x, y) and (y, x) will generically have different representations. Moreover, with reasonable assumptions we can generally recover the individual features of both x and y from this representation. If A and B are random orthogonal matrices, and we've assumed that the feature vectors $\{v_i\}$ are mutually close to orthogonal, it follows that the feature vectors $\{Av_1, \ldots, Av_m, Bv_1, \ldots, Bv_m\}$ will also be close to mutually orthogonal. The data for x and y can then be "read back" selectively, for example by reconstruction functions of the form

$$f_{A,i}(r) = \langle r, Av_i \rangle$$

$$f_{B,i}(r) = \langle r, Bv_i \rangle$$

Of course, there will be an increase in reconstruction error, since we are working with twice as many features. This issue may become significant if we move beyond pairs of vectors to encoding much longer lists. Thus the addition model of composition might put constraints on m, the number of individual feature vectors. On the positive side, the

assumptions on A, B, and the $\{v_i\}$ can be relaxed in various ways, and useful types of reconstruction may still be possible. For random matrices and most choices of v_i, these reconstruction functions will also work. If A and B are random but low-rank, full reconstruction won't be possible for all v_i, however it will be possible when $\|Av_i\|| \approx \|v_i\| = 1$.

This technique has been proposed in many different contexts. It is called an "additive model" in (Mitchell & Lapata, 2008; 2010), a "generic additive model" in (Zanzotto et al., 2010), and a closely related method is called "matrix binding of additive terms" in (Gallant & Okaywe, 2013). It is simple and flexible, naturally generalizing to represent ordered lists of size larger than two. It also has obvious parallels with the output of an attention layer of a transformer.

3.1. Matrix binding as writing to "slots" in superposition

There's an alternative perspective which relates directly to transformers. Consider an additive matrix binding model that combines a vector x with a list of vectors $(y_1, \ldots, y_k)$ via the representation:

$$r = x + A_1 y_1 + A_2 y_2 + \ldots + A_k y_k$$

One way to view this equation is that each matrix A_i defines a distinct "slot" where data can be written for later retrieval. In other words, the matrix binding formula augments the information in x with the information in the $\{y_i\}$, such that data in one slot can be distinguished from data in another.

By construction, attention layers perform something not unlike matrix binding[1]. There's a potentially fruitful connection between this setup and the proposal that the transformer residual stream is a communication channel (Elhage et al., 2021; Merullo et al., 2024) between different network layers, which can augment it with new data or read information from previous layers. It can be tempting to imagine that the residual stream S is a sum of vector spaces $S = E_1 \oplus E_2 \oplus \ldots, \oplus E_k$ where each E_i is a separate information "channel." However, the additive model is a reminder that channels themselves might be in superposition, with no clear geometric separation between them[2].

Finally, at a theoretical level, it's worth noting that the formula $Ax + By$ uses a form of superposition to create composition. As pointed out by Olah (Olah, 2024), in reviewing ideas of Thorpe (Thorpe, 1989), superposition and composition are typically in tension. In effect, the additive model is composing two dense codes by superposition. Of course, the slot-based model represents a slightly different type of "compositionality." But in Thorpe's terms one may view this as akin to a dense code with compositional characteristics, something in between a semi-distributed and highly distributed code[3].

3.2. Tree representations using matrices

Additive binding can be used to create data structures that are more complex than a collection of slots. For example, here's one possible way to represent a binary tree. Fix two random $n \times n$ matrices, M_1 and M_2. Consider a parent node p with two children represented by vectors c_1 and c_2. Then define the representation

$$r_p = M_1 c_1 + M_2 c_2$$

Applying this recursively, we can find a vector representing each node of the tree; and because generically $M_1 M_2 \neq M_2 M_1$, distinct nodes will be associated with a distinct vectors. While we have not found this mechanism explicitly mentioned in the literature, it can be viewed as a trivial linear version of a Tree-RNN, a network that creates a vector representation of a tree by operating recursively on its nodes (Socher et al., 2013; 2014; Bowman et al., 2015).

Tree-RNNs enjoyed a surge of interest in the years before transformers were introduced: using Tree-RNNs to process parse trees seemed to improve performance in multiple systems. Interestingly, for small fixed-depth trees there's a natural way for transformers to implement something like a Tree-RNN, since attention heads can focus on—and thus can read information from—syntactically dependent tokens (Phang et al., 2019; Ravishankar et al., 2021). It might be worthwhile to look for evidence of this type of representation in real-world networks.

3.3. Feature multiplicity in additive models

If neural networks use one of these matrix-based mechanisms, how might that affect the search for features? In an idealized case, there's actually a clear answer: it will create a series of duplicate features, which we call **echo features**.

To see this, consider a set of random unit feature vectors $\{v_1, \ldots, v_m\}$, and a model in which a representation vector x is a random sparse linear combination of these vectors. To be concrete, take $x = \sum_1^m a_i v_i$ where the a_i are each independent Bernoulli random variables with $p = 1/m$. Given a large sample of vectors from this distribution, a capable

[1] This analogy is suggestive, but not exact. Three important differences from classic matrix binding are nonlinearities (activation functions and layer norm), the low rank of individual attention output matrices, and the lack of guarantee that different output matrices are nearly orthogonal or far from the identity.

[2] Arguably, positional information in a transformer is encoded in just this way.

[3] One recent piece of empirical work is potentially related: (Engels et al., 2024) describe "feature manifolds," placed in superposition using linear transformations that satisfy a certain mutual orthogonality condition. This suggests that neural networks are capable of learning something like additive binding mechanisms.

dictionary learning algorithm, such as a sparse autoencoder, will recover the underlying feature vectors $\{v_i\}$.

Now consider a simple version of the additive model described above. Suppose we apply dictionary learning techniques to representations of the form $z = x + Ay$, where x and y are independently drawn from the same sparse random distribution. If A is random, the vectors $\{Av_i\}$ will probably be nearly orthogonal to the $\{v_i\}$. That means we can view $z = x + Ay$ as a sparse sum of $2m$ random vectors, where coefficients are independent Bernoulli with $p = 1/m$. Therefore, applying a dictionary learning algorithm to a large sample of z is likely to identify $2m$ feature vectors

$$\{v_1, \ldots, v_m, Av_1, \ldots, Av_m\}$$

The issue, of course, is that each pair of vectors (v_i, Av_i) actually corresponds to the same underlying feature—the only difference being its position in the ordered pair (x, y). We refer to multiple distinct vectors for the same concept as "echo" vectors since they echo each other's representation. Obviously, the problem is not just restricted to pairs: a representation of a list of three vectors would yield triple multiplicity. The tree representation described above could yield even more.

It's an open question whether this idealized setting reflects the behavior or real-world networks. However, the general structure of transformer attention suggests that something like this phenomenon could easily occur. In fact, one recent report describes how an SAE appears to uncover a large set of "induction features" corresponding to predicted next tokens (Kissane et al., 2024). A closely related issue is also documented in a real-world setting in (Makelov et al., 2023).

If this type of feature multiplicity does turn out to be a common phenomenon, it could make interpretability work more difficult. At the very least, it would greatly expand the catalog of seemingly independent features found by an SAE. Understanding the structure of this catalog would require knowledge of the underlying matrices that lead to echo vectors. Conversely, knowing these matrices would greatly help us organize and analyze SAE-derived features.

3.3.1. Feature multiplicity and predict/control discrepancies

Feature multiplicity might cause anomalies when using features to steer a network's behavior during inference. Indeed, a number of recent reports have described systems where the optimal vectors for predicting behavior and steering it turn out to be different (Zou et al., 2023; Li et al., 2024; Marks & Tegmark, 2023). We call this **predict/control discrepancy**. On its face, this discrepancy is not necessarily shocking—a linear probe can easily pick up on spurious or redundant correlations, such as non-causal aspects of the input data. In (Bricken et al., 2023), for example, the authors perform experiments to check that features have expected causal effects on the networks output. On the other hand, it is conceivable that feature multiplicity may make predict/control discrepancies especially likely[4].

Here's a hypothetical example that serves as a plausibility argument for this idea. Consider a transformer layer that does nothing but augment the residual stream for a given token with information about the previous token. Imagine it does so with additive matrix binding. In particular, if t_i and t_{i-1} are sequential tokens at layer $N-1$, then after layer N the residual stream at token i, which we denote by t'_i becomes

$$t'_i = t_i + At_{i-1}$$

where A is an orthogonal matrix far from the identity. Now imagine that representations in the residual stream for layer $N-1$ involve a feature vector v_{tox} for toxic content. It follows that, for different tokens, the residual stream in layer N could have multiple distinct vectors related to toxicity: v_{tox}, Av_{tox}, or $v_{tox} + Av_{tox}$. A linear probe P_{tox} trained to recognize toxic content, based on multiple tokens in the residual stream at layer N, might look something like

$$P_{tox}(x) = \langle x, v_{tox} + Av_{tox} \rangle$$

On the other hand, if we trained on just one token (perhaps the last token in a prompt, a non-toxic function word, in order to predict subsequent behavior) we could easily produce a probe such as:

$$Q_{tox}(x) = \langle x, Av_{tox} \rangle$$

Both of these probes might perform well, but this already illustrates a problem: different tokens might easily produce different probes for unimportant contextual reasons.

Now, suppose we wish to intervene during network inference—for instance, we want to change the residual stream to cause the network to produce less toxic text. Simply inspecting the second probe, we might hope to intervene to "remove toxicity" by intervening during inference (Li et al., 2024) to change each token t to have the new value

$$\tilde{t} = t - Av_{tox}$$

It's entirely possible, however, that the values of inner products $\langle x, v_{tox} \rangle$ and $\langle x, Av_{tox} \rangle$ will play different roles in computing the toxicity of the next word. It's certainly conceivable that the toxicity of the current word plays a greater role than information about the previous word. Quite likely, the most effective intervention will look more like the following equation, where $c_1 \neq c_2$:

$$\tilde{t} = t - c_1 v_{tox} - c_2 Av_{tox}$$

[4] See also (Marks, 2024) for a related discussion; we return to ideas from that essay in a subsequent section.

In other words, the right vector for *controlling* the network might differ from the most effective probe vector for *predicting* its behavior.

This phenomenon would make it harder to exploit features derived from dictionary learning. For example, if there are many multiple features related to toxicity, one couldn't automatically extract a steering vector from the dictionary, but would need to do experiments to find the right linear combination of feature vectors. Of course, the analysis outlined here is still speculative. On the other hand, we believe this is a plausible picture, especially given transformer architecture. A cynic might even say attention layers are machines for producing spurious correlations.

4. Multi-token mechanisms

So far, we've discussed binding mechanisms that operate wthin a single vector space. However, since the earliest days of the field, there has been speculation that that some notion of sequence—that is, representations across multiple input observations—is necessary for representing complex relationships (Rosenblatt, 1961). In this section, with the transformer architecture in mind, we discuss a class of mechanisms that relate specifically to sequence models.

Mathematically, we consider a sequence of "token vectors"

$$t_1, t_2, \ldots, t_k \in \mathbb{R}^n$$

where each t_i contains feature information and, potentially, information about its relation to other tokens. For instance, each t_i might represent the residual stream for the i-th element of a sequence at a particular layer in a transformer. To extract the relational data, it might not be enough to look at a single token t_i but at pairs of tokens (t_i, t_j). This perspective opens up a different set of possibilities. It's also a very plausible type of representation. For example, when working with word embeddings, a variety of relationships between words can famously be read by taking vector differences (Mikolov et al., 2013a).

A generalization of using vector differences is "linear relational embedding" (Paccanaro & Hinton, 2001). The idea here is that a relation between token vectors t_i and t_j is present when

$$t_i \approx At_j + b$$

for a matrix A and vector b which depend only on the relation. Recent work suggests that something like this representation may be found in the way that language-model transformers represent relationships between entities (Hernandez et al., 2023).

4.1. Syntactic relations and tree embeddings

One striking finding related to the BERT network (Devlin et al., 2018) is that parse trees are represented geometrically by relative positions of token vectors. In particular, the "syntactic distance" between two words—as measured in a dependency grammar parse tree—can be recovered from their corresponding token vectors in a middle layer of BERT (Hewitt & Manning, 2019; Manning et al., 2020). To be precise, if two words w_i, w_j in the same sentence are represented by embedding vectors t_i and t_j, and if $d_s(w_i, w_j)$ is the tree distance in the syntactic parse tree between w_i and w_2, then

$$\|Mt_i - Mt_j\|^2 \approx d_s(w_i, w_j)$$

where M is a constant linear transformation, depending only on the network. The fact that tree distance corresponds to the square of the Euclidean distance may seem surprising but, as described in (Reif et al., 2019), it's actually natural in the context of mapping a tree metric to a Euclidean metric. As described in (Chi et al., 2020), one possible explanation of this representation is that the differences between token embeddings represent syntactic dependencies: that is, if t_i and t_j have a specific syntactic relation (such as an adjective modifying a noun) then the vector difference $Mt_i - Mt_j$ encodes that relation.

This area seems like a promising direction for follow-up work. For one thing, tree structures might naturally occur in many other situations. More generally, the idea that differences between tokens encode specific, contextual relationships seems powerful. A natural question is whether applying dictionary learning to residual stream differences, rather than the residual stream itself, might yield a set of interpretable "relation features."

4.2. Reference mechanisms: pointers and identifiers

Software engineers have invented many ways to tie different data structures together. Is it possible that neural networks use the same techniques? Some recent investigations hint that they might.

One fundamental component of many software data structures is a **pointer**: a reference to a location in memory. The analog of a pointer, for a transformer network, might be a positional embedding that defines a reference to a specific token in a sequence. One might imagine using matrix binding to augment a token with this positional information rather than semantic data. For instance, if p_j represents the positional embedding for position j, and A_r is a binding matrix, one might represent a relationship between tokens t_i and t_j as:

$$r(t_i, t_j) = t_i + A_r p_j$$

In fact, the study described in (Prakash et al., 2024) uncovers a circuit that seems to use something like this kind of "pointer" mechanism to relate an entity to information about its state elsewhere in a sequence. The authors describe spe-

cific attention heads that seem to move and read relevant positional encodings in order to connect tokens in this way.

A second fundamental way to connect data comes from the world of databases: using a shared **identifier**, or ID, to link two pieces of information. Some transformers may actually use a geometric form of this mechanism (Feng & Steinhardt, 2023; Feng et al., 2024). The idea is that there may be a special subspace of the residual stream in which vector similarity encodes binding information. To be precise, they find a certain low-rank matrix A_{id}, with the property that for a token vector t, the value $A_{\mathrm{id}}t$ acts like an ID. That is, two token representations t_1 and t_2 are considered to be bound to each other if

$$A_{\mathrm{id}}t_i \approx A_{\mathrm{id}}t_j$$

One interesting question is how and whether this idea relates to the syntax tree representations described in (Manning et al., 2020). For example, after applying the proper structural probe, one could imagine projecting orthogonally to an adjective-modifies-noun direction. This could produce something like an ID vector linking adjectives with the nouns they modify.

A geometric ID mechanism may pose challenges for a pure feature-vector-based analysis. In particular, there's no reason to assume that ID information clusters in any useful way. The only structure that (Feng & Steinhardt, 2023) note in the subspace S is metric: nearby vectors are more likely to match as markers. Thus for any given sequence, ID vectors conceivably could be effectively random—the only constraint being that distinct ID vectors should be far from each other. Thus dictionary learning or classic linear probing may not encounter a useful signal.

5. Vector symbolic architecture

Finally, we discuss a set of historical ideas for vector binding, collectively known as "vector symbolic architecture," or VSA (Smolensky, 1990; Plate, 1994; 1997; Kanerva, 2009; Jones & Mewhort, 2007; Schlegel et al., 2022). A comprehensive survey and analysis of the zoo of techniques in this area is far beyond the scope of this paper. Our goal is simply to provide a gentle introduction to techniques that can potentially seem arcane, unmotivated, and intimidating. To do so, we focus on two foundational constructions, one in $\mathbb{R}^n$ and one in $\{0,1\}^n$, which underlie much of the work in this field. Both mechanisms have the potential to lead to feature "dark matter." That is, they might represent information in a way that is out of reach of dictionary learning or probing methods.

5.1. Tensor constructions

As described in (Schlegel et al., 2022), a technique introduced in (Smolensky, 1990) forms the basis of a large set of different VSA mechanisms. It is often referred to as a "tensor" method, but we'll describe it with the more down-to-earth notation of outer products.

As before, suppose we wish to compose, or bind, two vectors x and y. One way is to use an outer product to define a representation:

$$r = xy^T$$

This representation distinguishes order, since generally $xy^T \neq yx^T$). It respects vector addition, since it is linear in each of x and y respectively. Moreover, one can recover x and y, at least up to scalar multiples, with easy vector algebra. For example, if you know y, you can recover a scalar multiple of x via the product

$$ry = xy^Ty = \|y\|^2 x$$

The alert reader will notice, however, that this outer product representation is a cheat: it lives in $\mathbb{R}^{n\times n}$ rather than $\mathbb{R}^n$. One may view much of the VSA literature as a bag of tricks to get around this inconvenient fact. The most common trick is to create an n-dimensional projection of the outer product, which can still be used for approximate reconstruction. One method, for instance, is projecting the outer product matrix via circular convolution (Plate, 1997).

5.2. Binary vectors

A second category of VSA techniques relies on binary-valued vectors (Kanerva et al., 1997; Kanerva, 2009). We describe one simple proposal of this type. To bind two binary vectors x and y, we fix a permutation P that rearranges the entries of a vector[5], and then compute

$$r = x \oplus P(y)$$

Here $\oplus$ denotes modulo-2 addition. Like the outer product, this distinguishes between the ordered pairs (x, y) and (y, x). It respects the binary OR function, which can be used to take unions of two sets of features. It also allows for easy reconstruction of one vector in the pair if you know the other:

$$x = r \oplus P(y)$$

$$y = P^{-1}(r \oplus x)$$

This is a conceptually elegant method, although it requires extremely large vectors for effective storage and decoding.

[5]We need the permutation P because mod-2 addition is commutative, and otherwise we couldn't distinguish the pairs (x, y) and (y, x). The situation is similar to an additive matrix binding formula $x + Ay$. Permuting one argument also allows us to bind a vector to itself without leading to a zero representation.

5.3. Is VSA a plausible mechanism for real neural nets?

Unlike additive binding models, the techniques proposed for tensor-based VSA don't obviously map to deep learning architectures. Investigations of whether real-world networks use VSA have yielded mixed conclusions. For example, a study of whether CLIP might use these mechanisms produced largely negative results (Lewis et al., 2022), while there are some indications RNNs may reproduce tensor-related structures (McCoy et al., 2018).

Techniques based on binary vectors might seem even further from modern language model architectures. Surprisingly, some recent work suggests that for sparse vectors in superposition, not only can neural nets compute XORs of features in theory (Dmitry Vaintrob, 2024), they seem to do so in practice (Marks, 2024). Traditional binary VSA is based on dense vectors, of course, but this seems like an interesting avenue for exploration. One might imagine combining ideas from the tensor and binary VSA schemes—perhaps based on outer products specifically of zero-one vectors, either over $\mathbb{R}$ or $\mathbb{Z}_2$.

5.4. Implications for the search for features

As with matrix addition binding, we speculate that VSA mechanisms may lead to feature multiplicity. For example, outer product binding could plausibly produce a combinatorial proliferation of potential features[6]. To see the issue, consider a setting where we have m feature vectors, $\{v_1, \ldots, v_m\}$, and we want to compose representation vectors x and y:

$$x = \sum_{i=1}^{m} a_i v_i$$

$$y = \sum_{i=1}^{m} b_i v_i$$

where the sums are assumed to be sparse, and $\{a_i\}$ and $\{b_i\}$ independently chosen. Suppose we have a composition method of the form

$$r(x, y) = \pi(xy^T)$$

where $\pi : \mathbb{R}^{n \times n} \to \mathbb{R}^n$ is a projection operator. By linearity, we have

$$r(x, y) = \sum_{i,j} a_i b_j \pi(v_i v_j^T)$$

We have now moved from using m feature vectors $\{v_i\}$ to a situation with m^2 potential feature vectors $\{\pi(v_i v_j^T)\}$.

This is a problem. To begin with, even if we could use dictionary learning to find all these features, there may be a truly huge number of them. Even if not all combinations are likely to occur, the potential number of new combinatons is vast. For a person interpreting a network, having fine-grained versions of the same feature is confusing. For example, one might see many different vectors reflecting the same concept in different overall contexts. Something like this actually appears repeatedly in a language model analyzed in (Bricken et al., 2023). For example, they don't find a single pure "English preposition" feature, but they do identify a feature corresponding to "prepositions in scientific/statistical contexts"[7] and a separate feature that fires on "prepositions in contexts discussing poetry/poets"[8]. This is just one of many examples from the feature catalog of (Bricken et al., 2023). As the authors of that study discuss, this seems to be part of a much larger story related to a non-isotropic distribution of features—perhaps an analysis of compositional mechanisms could shed light on this phenomenon.

A second problem with the proliferation of fine-grained features is that we might need far more data to discover each one. For example, features for "inspiring" and "architecture" might be just common enough to find, but the combination of "inspiring architecture" might, sadly, be too rare to appear in the output of a dictionary learning algorithm. This may lead to "dark matter"—feature representation that are out of reach unless one tests with vast amounts of data.

From an optimistic viewpoint, this phenomenon can also be seen as a cause for hope. As discussed earlier, many features in the catalog of (Bricken et al., 2023) seem to have combinatorial interpretations. If we had a principled way of "factoring" these features, extracting an outer-product structure automatically, that could be a huge help in identifying primitive features. It could also help simplify and organize feature catalogs.

6. Conclusion and future work

We've described a series of proposals—some historical, some very recent—for how neural nets might represent relationships between features. Along the way, we've given conceptual arguments that these mechanisms might present problems when finding and using linear feature representations. One challenge is that of "dark matter": feature representations that are difficult to find using standard methods. Another potential problem is feature multiplicity, or the presence of multiple "echo vectors" that correspond to the same feature. An issue of special concern, related to multiplicity, is whether feature vectors discovered by probing or dictionary-learning methods will be as useful for interventions as they are for predicting the state of the network.

[6] The essay (Marks, 2024) has a discussion of parallel concerns in the case of binary codes.

[7] Feature A/1/3533 in the study's notation

[8] Feature A/1/3977 in the study's notation

The arguments here are conjectural and often involve idealized settings. A great deal of work would be necessary to resolve the questions we've raised—calling for that work is the point of this note. Here are some directions that might shed light on the key issues:

- **Toy models that learn composition**. One way to gather data would be to experiment with minimal models. It would be useful to find toy models of learned composition. If we give an autoencoder a task that requires learning relational composition, what mechanisms are found via gradient descent? Some work in this area dates back decades (Pollack, 1988; Blank et al., 2014) and it would be interesting to analyze these same systems with modern methods, and to extend this line of research further.
- **Apply feature extraction methods to synthetic composition mechanisms**. We provided a conceptual outline of how feature discovery techniques might run into trouble on compositional representations. A natural next step would be to test these arguments empirically, by creating synthetic versions of VSA, matrix binding, and other mechanisms, and applying dictionary learning to the results.
- **Apply dictionary learning to token differences**. Syntactic relations appear to be encoded in differences between token vectors. Are other relations encoded this way as well? It could be worth applying dictionary learning techniques to token differences to find additional "relational features."
- **Investigate marker mechanisms**. The "ID vector" mechanism suggested by (Feng & Steinhardt, 2023) is extremely interesting. It also presents a clear problem for feature-based interpretability methods. It would be helpful to understand just how widely the mechanism might apply. How does it relate to the syntax representations of (Manning et al., 2020)? Experimentation with small synthetic models might advance our understanding of the basic process. It would also be helpful to find lightweight ways to identify ID vector subspaces.
- **Look for feature multiplicity**. If feature multiplicity is a real problem, it may not be hard to identify. One sign would be the discovery of redundant or contextually dependent feature vectors. The features found in (Bricken et al., 2023; Templeton et al., 2024; Kissane et al., 2024) sometimes exhibit this type of behavior, for instance. If one could pair up a sufficiently large number of redundant features via a single linear transformation, that would provide strong evidence for the multiplicity hypothesis, and also help organize feature catalogs.
- **Understand predict/control discrepancy**. Feature multiplicity might cause a problem that we have termed "predict/control discrepancy." That is, an effective vector for predicting a particular network behavior might be different from the best "steering vector" for inducing that same behavior. Given the obvious practical implications, and that there are multiple examples of this phenomenon in the literature, it seems important to investigate further.
- **Measure real models for relational composition**. One alternative to explicitly identifying relational representations is to look for behavioral evidence that they exist. That can at least alert us to "dark matter" that isn't observed by feature identification methods. A review of this type of work is beyond the scope of this paper, but several studies suggest promising directions (Andreas, 2019; Lovering & Pavlick, 2022; Akyürek & Andreas, 2023).
- **Investigate VSA-based binding in real networks.** The findings in (Dmitry Vaintrob, 2024; Marks, 2024) suggest some natural mechanisms by which binary VSA codes might be implemented in realistic networks. It may be worth systematically working out the simplest implementations of these mechanisms, and investigating whether real networks make use of them.

To sum up, there are major practical and theoretical questions around how the linear representation hypothesis might interact with mechanisms for relational composition. We strongly advocate for continued research in this area. Identifying mechanisms for relational composition would be a step forward for useful interpretability work. Even if it turns out that neural nets show no evidence of relational composition, that would be an important theoretical result.

7. Acknowledgments

We thank Kenneth Li, Oam Patel, Nikola Jurkovic, Yonatan Belinkov, Asma Ghandeharioun, Ann Yuan, and Lucas Dixon for helpful comments on this manuscript. We're indebted to the anonymous reviewers for their careful reading and excellent suggestions. FV was supported by a fellowship from the Radcliffe Institute for Advanced Study at Harvard University. Additional support came from Effective Ventures Foundation, Effektiv Spenden Schweiz, and the Open Philanthropy Project.

References

Akyürek, E. and Andreas, J. Lexsym: Compositionality as lexical symmetry. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 639–657, 2023.

Andreas, J. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

Blank, D. S., Meeden, L. A., and Marshall, J. B. Exploring the symbolic/subsymbolic continuum: A case study of raam. In *The Symbolic and Connectionist Paradigms*, pp. 113–148. Psychology Press, 2014.

Bowman, S., Potts, C., and Manning, C. D. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pp. 12–21, 2015.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, pp. 2, 2023.

Chi, E. A., Hewitt, J., and Manning, C. D. Finding universal grammatical relations in multilingual bert. *arXiv preprint arXiv:2005.04511*, 2020.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dmitry Vaintrob, Jake Mendel, K. Toward a mathematical framework for computation in superposition. *LessWrong*, 2024. URL https://www.lesswrong.com/posts/2roZtSr5TGmLjXMnT/toward-a-mathematical-framework-for-computation-in-s

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1, 2021.

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

Engels, J., Liao, I., Michaud, E. J., Gurnee, W., and Tegmark, M. Not all language model features are linear. *arXiv preprint arXiv:2405.14860*, 2024.

Feldman, J. The neural binding problem (s). *Cognitive neurodynamics*, 7:1–11, 2013.

Feng, J. and Steinhardt, J. How do language models bind entities in context? *arXiv preprint arXiv:2310.17191*, 2023.

Feng, J., Russell, S., and Steinhardt, J. Monitoring latent world states in language models with propositional probes. *arXiv preprint arXiv:2406.19501*, 2024.

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

Gallant, S. I. and Okaywe, T. W. Representing objects, relations, and sequences. *Neural computation*, 25(8): 2038–2078, 2013.

Hernandez, E., Sharma, A. S., Haklay, T., Meng, K., Wattenberg, M., Andreas, J., Belinkov, Y., and Bau, D. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*, 2023.

Hewitt, J. and Manning, C. D. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

Jones, M. N. and Mewhort, D. J. Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, 114(1):1, 2007.

Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1: 139–159, 2009.

Kanerva, P. et al. Fully distributed representation. *PAT*, 1 (5):10000, 1997.

Kissane, C., Robertzk, Conmy, A., and Nanda, N. Sparse autoencoders work on attention layer outputs. https://www.lesswrong.com/posts/DtdzGwFh9dCfsekZZ/sparse-autoencoders-work-on-attention-layer-outpu 2024. Accessed: 2024-07-13.

Lewis, M., Nayak, N. V., Yu, P., Yu, Q., Merullo, J., Bach, S. H., and Pavlick, E. Does clip bind concepts? probing compositionality in large image models. *arXiv preprint arXiv:2212.10537*, 2022.

Li, K., Patel, O., Viégas, F., Pfister, H., and Wattenberg, M. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36, 2024.

Lovering, C. and Pavlick, E. Unit testing for concepts in neural networks. *Transactions of the Association for Computational Linguistics*, 10:1193–1208, 2022.

Makelov, A., Lange, G., Geiger, A., and Nanda, N. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. In *The Twelfth International Conference on Learning Representations*, 2023.

Manning, C. D., Clark, K., Hewitt, J., Khandelwal, U., and Levy, O. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

Marks, S. What's up with llms representing xors of arbitrary features? *LessWrong*, 2024. URL https://www.lesswrong.com/posts/hjJXCn9GsskysDceS/what-s-up-with-llms-representing-xors-of-arbitrary-features.

Marks, S. and Tegmark, M. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

McCoy, R. T., Linzen, T., Dunbar, E., and Smolensky, P. Rnns implicitly implement tensor product representations. *arXiv preprint arXiv:1812.08718*, 2018.

Merullo, J., Eickhoff, C., and Pavlick, E. Talking heads: Understanding inter-layer communication in transformer language models. *arXiv preprint arXiv:2406.09519*, 2024.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013a.

Mikolov, T., Yih, W.-t., and Zweig, G. 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*, pp. 746–751, 2013b.

Mitchell, J. and Lapata, M. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT*, pp. 236–244, 2008.

Mitchell, J. and Lapata, M. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

Olah, C. Distributed representations: Composition superposition. *Transformer Circuits*, 2024. URL https://transformer-circuits.pub/2023/superposition-composition.

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

Paccanaro, A. and Hinton, G. E. Learning distributed representations of concepts using linear relational embedding. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):232–244, 2001.

Phang, J., Bordia, S., Bowman, S. R., et al. Do attention heads in bert track syntactic dependencies? In *NY Academy of Sciences NLP, Dialog, and Speech Workshop*, 2019.

Plate, T. A common framework for distributed representation schemes for compositional structure. *Connectionist systems for knowledge representation and deduction*, pp. 15–34, 1997.

Plate, T. A. *Distributed representations and nested compositional structure*. Citeseer, 1994.

Pollack, J. Implications of recursive distributed representations. *Advances in neural information processing systems*, 1, 1988.

Prakash, N., Shaham, T. R., Haklay, T., Belinkov, Y., and Bau, D. Fine-tuning enhances existing mechanisms: A case study on entity tracking. *arXiv preprint arXiv:2402.14811*, 2024.

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

Ravishankar, V., Kulmizev, A., Abdou, M., Søgaard, A., and Nivre, J. Attention can reflect syntactic structure (if you let it). *arXiv preprint arXiv:2101.10927*, 2021.

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., and Kim, B. Visualizing and measuring the geometry of bert. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Schlegel, K., Neubert, P., and Protzel, P. A comparison of vector symbolic architectures. *Artificial Intelligence Review*, 55(6):4523–4555, 2022.

Smolensky, P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

Socher, R., Bauer, J., Manning, C. D., and Ng, A. Y. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 455–465, 2013.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

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

Thorpe, S. Local vs. distributed coding. *Intellectica*, 8(2): 3–40, 1989.

Zanzotto, F., Korkontzelos, I., Fallucchi, F., Manandhar, S., et al. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd international conference on computational linguistics (COLING)(GGS Conference Rating 2 A)*, 2010.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A. Appendix: Features

In this paper we use the word "feature" formally, as a kind of mathematical abstraction, since that's all that's needed for our arguments. However, it's worth talking about some of the intuition behind the word. To begin with, there's not a consensus definition in the literature, which is too large to survey in an appendix. See (Elhage et al., 2022), for example, which discusses three approaches: features as arbitrary functions of the input, features as interpretable properties, and—most abstractly—features as what an infinitely large neural net might devote a single neuron to.

The general intuition we've found most helpful, however, is simply that a feature represents a unit of data that is useful for future computations. We also believe it's helpful to divide these representations into three broad categories. Some features relate to the input—that is, they represent a useful property of the data seen by the network. ("Red area on green background" or "apple.") Other features, like data structures in a traditional algorithm, are related to intermediate processing. ("Food and hunger".) Finally, a third set of features will relate to the result of the network's computations: they will represent properties of the output, rather than input ("Reach arm" or "open mouth").

The analogy is obviously with biological nervous systems, which can be broken into areas for sensory, motor, and internal processing. Of course, there's no guarantee that a neural network maintains a strict division between these different aspects. It's conceivable that within a given layer, all three types of features may be found. That may be another reason why interpreting features can be hard. Frequently people try to understand features in terms of input data, when those features may really be aspects of hidden computation or output. It may also explain some predict/control discrepancies: linear probes may be effectively trained in a way that emphasizes features of the input, which are only roughly correlated with output features.