

The Strategic Picture of Superposition

Although superposition is scientifically interesting, much of our interest comes from a pragmatic motivation: we believe that superposition is deeply connected to the challenge of using interpretability to make claims about the safety of AI systems. In particular, it is a clear challenge to the most promising path we see to be able to say that neural networks won't perform certain harmful behaviors or to catch "unknown unknowns" safety problems. This is because superposition is deeply linked to the ability to identify and enumerate over all features in a model, and the ability to enumerate over all features would be a powerful primitive for making claims about model behavior.

We begin this section by describing how "solving superposition" in a certain sense is equivalent to many strong interpretability properties which might be useful for safety. Next, we'll describe three high level strategies one might take to "solving superposition." Finally, we'll describe a few other additional strategic considerations.

Safety, Interpretability, & "Solving Superposition"

We'd like a way to have confidence that models will never do certain behaviors such as "deliberately deceive" or "manipulate." Today, it's unclear how one might show this, but we believe a promising tool would be the ability to *identify and enumerate over all features*. The ability to have a universal quantifier over the fundamental units of neural network computation is a significant step towards saying that certain types of circuits don't exist.¹⁸ It also seems like a powerful tool for addressing "unknown unknowns", since it's a way that one can fully cover network behavior, in a sense.

How does this relate to superposition? It turns out that the ability to enumerate over features is deeply intertwined with superposition. One way to see this is to imagine a neural network with a privileged basis and without superposition (like the monosemantic neurons found in early InceptionV1, e.g. [1]): features would simply correspond to neurons, and you could enumerate over features by enumerating over neurons.¹⁹ The connection also goes the other way: if one has the ability to enumerate over features, one can perform compressed sensing using the feature directions to (with high probability) "unfold" a superposition models activations into those of a larger, non-superposition model.

For this reason, we'll call any method that gives us the ability to enumerate over features – and equivalently, unfold activations – a "solution to superposition". Any solution is on the table, from creating models that just don't have superposition, to identifying what directions correspond to features after the fact. We'll discuss the space of possibilities shortly.

We've motivated "solving superposition" in terms of feature enumeration, but it's worth noting that it's equivalent to (or necessary for) many other interpretability properties one might care about:

- **Decomposing Activation Space.** The most fundamental challenge of any interpretability agenda is to defeat the curse of dimensionality. For mechanistic interpretability, this ultimately reduces to whether we can decompose activation space into independently understandable components, analogous to how computer program memory can be decomposed into variables. Identifying features is what allows us to decompose the model in terms of them.
- **Describing Activations in Terms of Pure Features.** One of the most obvious casualties of superposition is that we can't describe activations in terms of pure features. When features are relatively basis aligned, we can take an activation – say the activations for a dog head in a vision model – and decompose them into individual underlying features, like a floppy ear, short golden fur, and a snout. (See the "semantic dictionary" interface in [Building Blocks](#) [37].) Solving superposition would allow us to do this for every model.
- **Understanding Weights (ie. Circuit Analysis).** Neural network weights can typically only be understood when they're connecting together understandable features. All the circuit analysis seen in the original circuit thread (see especially [38]), see specially was fundamentally only possible because the weights connected non-polysemantic neurons. We need to solve superposition for this to work in general.
- **Even very basic approaches become perilous with superposition.** It isn't just sophisticated approaches to interpretability which are harmed by superposition. Even very basic methods one might consider become unreliable. For example, if one is concerned about language models exhibiting manipulative behavior, one might ask if an input has a significant cosine similarity to the representations of other examples of deceptive behavior. Unfortunately, superposition means that cosine similarity has the potential to be misleading, since unrelated features start to be embedded with positive dot products to each other. However, if we solve superposition, this won't be an issue – either we'll have a model where features align with neurons, or a way to use compressed sensing to lift features to a space where they no longer have positive dot products.

Three Ways Out

At a very high level, there seem to be three potential approaches to resolving superposition:

- **Create models without superposition.**
- **Find an overcomplete basis** that describes how features are represented in models with superposition.
- **Hybrid approaches** in which one changes models, not resolving superposition, but making it easier for a second stage of analysis to find an overcomplete basis that describes it.

Our sense is that all of these approaches are possible if one doesn't care about having a competitive model. For example, we believe it's possible to accomplish any of these for the toy models described in this paper. However, as one starts to consider serious neural networks, let alone modern large language models, all of these approaches begin to look very difficult. We'll outline the challenges we see for each approach in the following sections.

With that said, it's worth highlighting one bright spot before we focus on the challenges. You might have believed that superposition was something you could never fully get rid of, but that doesn't seem to be the case. All our results seem to suggest that superposition and polysemanticity are phases with sharp transitions. That is, there may exist a regime for every model where it has no superposition or polysemanticity. The question is largely whether the cost of getting rid of or otherwise resolving superposition is too high.

APPROACH 1: CREATING MODELS WITHOUT SUPERPOSITION

It's actually quite easy to get rid of superposition in the toy models described in this paper, albeit at the cost of a higher loss. Simply apply an L1 regularization term to the hidden layer activations (i.e. add $\lambda \|h\|_1$ to the loss). This actually has a nice interpretation in terms of killing features below a certain importance threshold, especially if they're not basis aligned. Generalizing this to real neural networks isn't trivial, but we expect it can be done.

However, it seems likely that models are significantly benefitting from superposition. Roughly, the sparser features are, the more features can be squeezed in per neuron. And many features in language models seem very sparse! For example, language models know about individuals with only modest public presences, such as several of the authors of this paper. Presumably we only occur with frequency significantly less than one in a million tokens. As a result, it may be the case that superposition effectively makes models much bigger.

All of this paints a picture where getting rid of superposition may be fairly achievable, but doing so will have a large performance cost. For a model with a fixed number of neurons, superposition helps – potentially a lot.

But this is only true if the constraint is thought of in terms of neurons. That is, a superposition model with n neurons likely has the same performance as a significantly larger monosemantic model with kn neurons. But neurons aren't the fundamental constraint: flops are. In the most common model architectures, flops and neurons have a strict correspondence, but this doesn't have to be the case and it's much less clear that superposition is optimal in the broader space of possibilities.

One family of models which change the flop-neuron relationship are Mixture of Experts (MoE) models (see [review \[39\]](#)). The intuition is that most neurons are for specialized circumstances and don't need to activate most of the time. For example, German-specific neurons don't need to activate on French text. Harry Potter neurons don't need to activate on scientific papers. So MoE models organize neurons into blocks or experts, which only activate a small fraction of the time. This effectively allows the model to have k times more neurons for a similar flop budget, given the constraint that only $1/k$ of the neurons activate in a given example and that they must activate in a block. Put another way, MoE models can recover neuron sparsity as free flops, as long as the sparsity is organized in certain ways.

It's unclear how far this can be pushed, especially given difficult engineering constraints. But there's an obvious lower bound, which is likely too optimistic but is interesting to think about: what if models only expended flops on neuron activations, and recovered the compute of all non-activating neurons? In this world, it seems unlikely that superposition would be optimal: you could always split a polysemantic neuron into dedicated neurons for each feature with the same cost, except for the cases where there would have been interference that hurt the model anyways. Our preliminary investigations comparing various types of superposition in terms of "loss reduction per activation frequency" seem to suggest that superposition is not optimal on these terms, although it asymptotically becomes as good as dedicated feature dimensions. Another way to think of this is that superposition exploits a gap between the sparsity of neurons and the sparsity of the underlying features; MoE eats that same gap, and so we should expect MoE models to have less superposition.

To be clear, MoE models are already well studied, and we don't think this changes the capabilities case for them. (If anything, superposition offers a theory for why MoE models have not proven more effective for capabilities when the case for them seems so initially compelling!) But if one's goal is to create competitive models that don't have superposition, MoE models become interesting to think about. We don't necessarily think that they specifically are the right path forward – our goal here has been to use them as an example of why we think it remains plausible there may be ways to build competitive superposition-free models.

APPROACH 2: FINDING AN OVERCOMPLETE BASIS

The opposite strategy of creating a superposition-free model is to take a regular model, which has superposition, and find an overcomplete basis describing how features are embedded after the fact.