**ReLU Hidden Layer Toy Model on Absolute Value Task**
$n = 100;\ m = 40;\ I_i = 0.8^i$


$1 - S = 1.0$

In the dense regime, all neurons are monosemantic, dedicated to a single feature.

$1 - S = 0.3$

Neruons continue to be monosemantic to moderate sparsity levels.

$1 - S = 0.1$

Eventually, we start to see a few slightly polysemantic neurons.

$1 - S = 0.03$

As sparsity increases further, we see a small number of highly polysemantic neurons representing low importance features.

$1 - S = 0.01$

The number of polysemantic neurons grows...

$1 - S = 0.003$

And they become even more polysemantic...

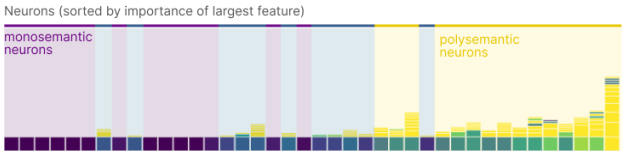
$1 - S = 0.001$

Eventually, all neurons are highly polysemantic.

Much like we saw in the ReLU hidden layer models, these results demonstrate that activation functions, under the right circumstances, create a privileged basis and cause features to align with basis dimensions. In the dense regime, we end up with each neuron representing a single feature, and we can read feature values directly off of neuron activations.

However, once the features become sufficiently sparse, this model, too, uses superposition to represent more features than it has neurons. This result is notable because it demonstrates the ability of neural networks to **perform computation** even on data that is represented in superposition.[17] Remember that the model is required to use the hidden layer ReLU in order to compute an absolute value; gradient descent manages to find solutions that usefully approximate the computation even when each neuron encodes a mix of multiple features.

Focusing on the intermediate sparsity regimes, we find several additional qualitative behaviors that we find fascinatingly reminiscent of behavior that has been observed in real, full-scale neural networks:




$1 - S = 0.01$

Neurons can be monosemantic and polysemantic in the same model.

To begin, we find that in some regimes, **many** of the model's neurons will encode pure features, but a subset of them will be highly polysemantic. This is similar to the phase change we saw earlier in the Relu output model. However, in that case, the phase change was with respect to features, with more important features not being put in superposition. In this experiment, the neurons don't have any intrinsic importance, but we see that the neurons representing the most important features (on the left) tend to be monosemantic.

We find this to bear a suggestive resemblance to some previous work in vision models, which found some layers that contained "mostly pure" feature neurons, but with some neurons representing additional features on a different scale.

We also note that many neurons appear to be associated with a single "primary" feature – encoded by a relatively large weight – coupled with one or more "secondary" features encoded with smaller-magnitude weights to that neuron. If we were to observe the activations of such a neuron over a range of input examples, we would find that the largest activations of that neuron were all or nearly-all associated with the presence of the "primary" feature, but that the lower-magnitude activations were much more polysemantic.

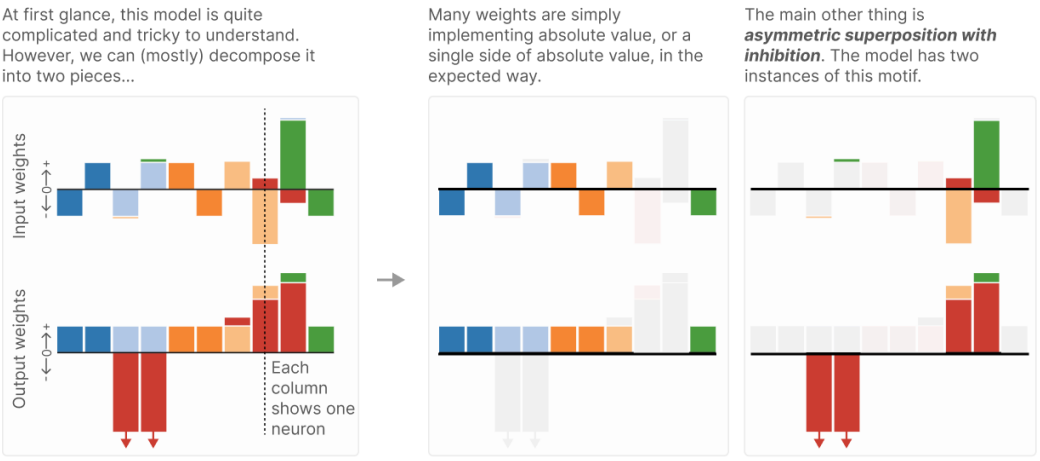
Intriguingly, that description closely matches what researchers have found in previous work on language models [2] – many neurons appear interpretable when we examine their strongest activations over a dataset, but can be shown on further investigation to activate for other meanings or patterns, often at a lower magnitude. While only suggestive, the ability of our toy model to reproduce these qualitative features of larger neural networks offers an exciting hint that these models are illuminating general phenomena.

## The Asymmetric Superposition Motif

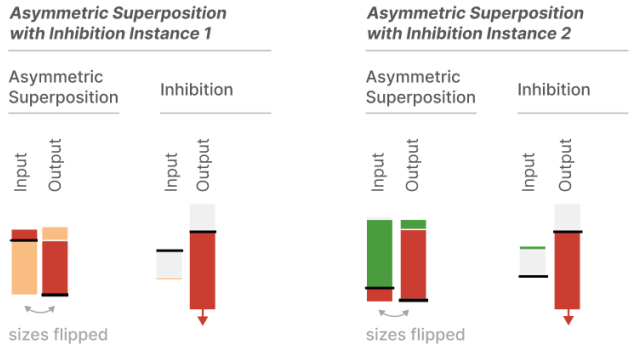
If neural networks can perform computation in superposition, a natural question is to ask how exactly they're doing so. What does that look like mechanically, in terms of the weights? In this subsection, we'll (mostly) work through one such model and see an interesting motif of ***asymmetric superposition***. (We use the term "motif" in the sense of the original circuit thread, inspired by its use in systems biology [36].)

The model we're trying to understand is shown below on the left, visualized as a neuron weight stack plot, with features corresponding to colors. The model is only doing a limited amount of superposition, and many of the weights can be understood as simply implementing absolute value in the expected way.

However, there are a few neurons doing something else...

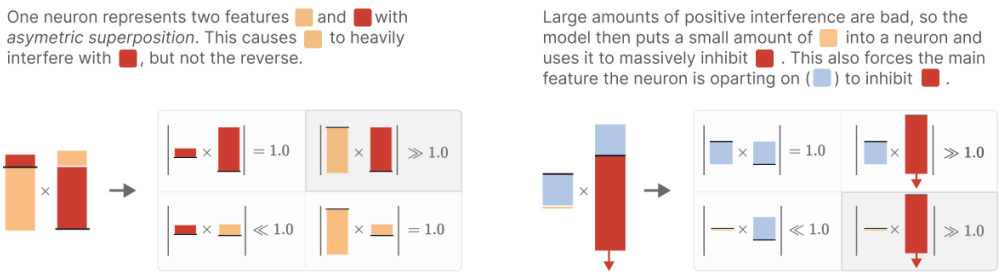



These other neurons implement two instances of asymmetric superposition and inhibition. Each instance consists of two neurons:




One neuron does *asymmetric superposition*. In normal superposition, one might store features with equal weights (eg. $W = [1, -1]$) and then have equal output weights ($W = [1, 1]$). In asymmetric superposition, one stores the features with different magnitudes (eg. $W = [2, -\frac{1}{2}]$) and then has reciprocal output weights (eg. $W = [\frac{1}{2}, 2]$). This causes one feature to heavily interfere with the other, but avoid the other interfering with the first!

To avoid the consequences of that interference, the model has another neuron heavily inhibit the feature in the case where there would have been positive interference. This essentially converts positive interference (which could greatly increase the loss) into negative interference (which has limited consequences due to the output ReLU).




There are a few other weights this doesn't explain. (We believe they're effectively small conditional biases.) But this asymmetric superposition and inhibition pattern appears to be the primary story.