Investigating Polysemantic Neurons in the MLP of Transformers

Samuel Borovoy

December 2024

Abstract

This paper explores the phenomena of superposition and polysemanticity in neural networks, which complicate the interpretability of how features are represented in the model. I investigate the behavior of features in the activation space of neural networks, particularly within the MLP blocks of a Transformer. Features are often represented by combinations of activations across multiple neurons, making it difficult to disentangle individual features. Superposition enables neural networks to represent more features than the number of neurons, showcasing one of their greatest strengths. However, this comes at the expense of interpretability, often leading them to be viewed as "black boxes." The experiments in this paper reveal that as input sparsity increases, the nonlinear model begins to represent more features than its dimensionality, leading to polysemantic behavior where neurons encode multiple features.

1 Introduction

The goal of this paper is to investigate superposition and polysemanticity in neural networks – two phenomena that complicate our ability to interpret how these models represent features. Intuitively, one might expect a neural network to store each feature (some distinct idea or concept, e.g., Prince Charles, Basketball, Photosynthesis, etc.) in a single, dedicated neuron. However, in practice, neural networks rarely exhibit such clean, one-to-one relationships. Instead, features are often represented as combinations of activations across multiple neurons. This property, known as superposition, means that the *same* neuron can participate in encoding multiple unrelated or overlapping features. When a single neuron exhibits this behavior, it is said to be polysemantic. Superposition and polysemanticity make it challenging to understand the "black box" of neural networks because features don't neatly align with a singular dimension. Gaining insight into the conditions that facilitate these properties is crucial for improving interpretability, an hot issue in AI research at the moment.

To empirically investigate superposition, the Multilayer Perceptron (MLP) block of a Transformer serves as a natural starting point. While Transformers

are often celebrated for their attention mechanisms, the MLP block accounts for approximately two-thirds of the model's total parameters[1]. As a relatively simple dense feed forward neural network, the MLP block provides a suitable environment for studying superposition, for reasons that will be elaborated upon later.

2 Mathematical Setup

In a perfectly interpretable setting, neural networks are thought of as having features of the input represented as directions in activation space, where features are learned during the training process. For example, in this high dimensional activation space (where V(W) is the vector representing some word in high dimensional space):

$$V(\text{"King"}) - V(\text{"Man"}) + V(\text{"Woman"}) = V(\text{"Queen"}).$$

This result[2] shows that some concept of gender is understood by the word representations in vector space, proving the direction of these vectors do embed some meaning. To clarify, a neural network layer with N neurons will have a N-dimensional activation space where the activations of these N neurons for a given input are represented as vectors in N-dimensional space.

A feature, f, has an activation that can be described as:

$$Activation(f) = X_f \cdot W_f$$

where X represents the matrix of inputs and W is the matrix of weights.

This optimistic approach to interpretability has the following two assumptions:

1. *Decomposability:* Network representations can be described in terms of independently understandable features.

2. *Linearity:* Features are represented by direction.

However, the subject of this paper seeks to understand these two conflicting properties for why neurons may or may not directly map to a given feature:

1. *Privileged Basis:* Only some networks have a privileged basis, which encourages features aligning with basis directions.

2. *Superposition:* Neural networks can represent more features than dimensions. This can be seen as neural networks simulating larger networks, where features don't directly correspond to individual neurons.

[1] How Might LLMs Store Facts? `https://youtu.be/9-Jl0dxWQs8?si=5BOu26gNvM0FHt1i`

[2] Efficient Estimations of Word Representations in Vector Space: `https://arxiv.org/pdf/1301.3781`

It is the ReLU activation functions in a neural network that encourages a privileged basis. ReLU sets negative inputs to zero, resulting in highly sparse activations. Consequently, only a smaller subset of neurons contribute to the output, effectively emphasizing specific directions in the activation space. By selectively activating certain neurons, ReLU enables the network to associate distinct features with specific basis directions, thereby making these directions privileged. Hence, one might expect features to align with the N basis dimensions in an interpretable way. However, a rich model might want to capture more features than dimensions, so the network tends to represent features in a way that is "almost orthogonal." In fact, in the kinds of high-dimensional spaces that exist in neural networks, there can exist $\exp(N)$ almost orthogonal vectors in an activation space, assuming there are N dimensions, by the Johnson-Lindenstrauss Lemma.

This is an important property of high-dimensional spaces. This can be imagined as a neural network in N-dimensions simulating a larger N'-dimensional network that has every neuron representing a distinct, disentangled feature, where $N' > N$.

3 Nonlinear Networks and Superposition

Researchers at Anthropic conducted an experiment[3] to investigate the conditions under which superposition arises. They trained a linear model and a nonlinear model to see when superposition would be exhibited. The two models looked as follows:

Linear Model	**ReLU Output Model**
$h = Wx$	$h = Wx$
$x' = W^T h + b$	$x' = \text{ReLU}(W^T h + b)$
$x' = W^T W x + b$	$x' = \text{ReLU}(W^T W x + b)$

Figure 1: Linear and Nonlinear Models

The goal of the experiment was to explore whether a neural network can project a high-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ into a lower-dimensional vector $\mathbf{h} \in \mathbb{R}^m$ and then recover it. To recover the original vector from the lower-dimensional representation $\mathbf{h}$, the researchers applied $\mathbf{W}^\top$.

This approach of using $\mathbf{W}^\top$ is considered "relatively mathematically principled" and has been shown to work effectively in practice. After training the models, the important questions to be answered are:
1. How does linearity affects how many features each model learned?
2. How much does the amount of superposition vary in each model type?

[3]Toy Models of Superposition: `https://transformer-circuits.pub/2022/toy_model/index.html`

For any feature, whether or not it is represented is determined by $\|\mathbf{W}_i\|$, the norm of its embedding vector.

To assess whether a feature shares a dimension with other features, the researchers used the metric:

$$\sum_{j \neq i} (\mathbf{W}_i \cdot \mathbf{W}_j)^2$$

If features are embedded in different dimensions, the dot product $(\mathbf{W}_i \cdot \mathbf{W}_j)$ is 0, as the vectors are orthogonal. On the other hand, values ≥ 1 indicate that there exists some group of other features that can activate $\mathbf{W}_i$ as strongly as feature i itself. Hence, the larger the sum value, the more superposition in the model.

In networks trained on n features with m dimensions where $n > m$, the Anthropic paper found 3 important results:

1. Linear models learned the top m most important features, and did not have superpositions.
2. With dense inputs, the ReLU model also learned the top m most important features, and did not have superpositions.
3. As sparsity of inputs increases, the ReLU model begins to represent more than m features.

Result 1 can be explained by the fact that a linear model is really just a neural network without activation functions, essentially a complex linear function. The linear model can be thought of optimizing some loss function where I is a function of feature importance:

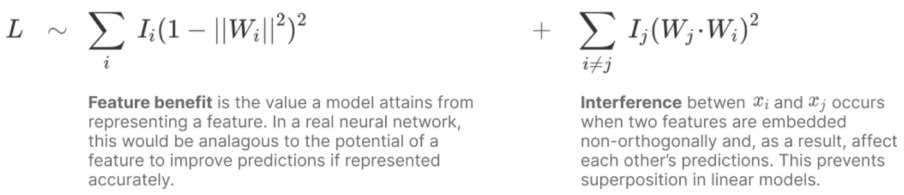

Figure 2: Learning Dynamics in Linear Models
[4]

Figure 2 shows the key tradeoff in the learning process for a linear model. The model can reduce loss by representing more features (feature benefit), but

it can only do so at the cost of increasing loss (feature interference). This relationship makes it never optimal to represent more features than dimensions in the network.

Result 2 can be explained by the ReLU model minimizing this loss function:

$$L = \int_x \left\| I(x - \text{ReLU}(W^T W x + b)) \right\|^2 \, dp(x) \tag{1}$$

where x is distributed such that $x_i = 0$ with probability S.

This loss can be thought of minimizing the expected loss over the distribution of inputs, meaning the model is trained to minimize the discrepancy between the original inputs and the predicted outputs across the entire space of possible inputs, weighted by their likelihood dp(x).

The above integral decomposes to:

$$L = (1-S)^n L_n + ... + (1-S)^{S^{n-1}} L_1 + S^n L_0, \tag{2}$$

with each L_k corresponding to the loss when the input is a k-sparse vector.

Note that as $S \to 1$, L_1 and L_0 dominate. The L_0 term, corresponding to the loss on a zero vector, which just is a positive bias. The interesting term is L_1, the loss on 1-sparse vectors.

$$L_1 = \sum_i \int_{0 \le x_i \le 1} I_i (x_i - \text{ReLU}(||W_i||^2 x_i + b_i))^2 + \sum_{i \neq j} \int_{0 \le x_i \le 1} I_j \text{ReLU}(W_j \cdot W_i x_i + b_j)^2$$

If we focus on the case $x_i = 1$, we get something which looks even more analagous to the linear case:

$$= \sum_i I_i (1 - \text{ReLU}(||W_i||^2 + b_i))^2 + \sum_{i \neq j} I_j \text{ReLU}(W_j \cdot W_i + b_j)^2$$

Feature benefit is similar to before. Note that ReLU never makes things worse, and that the bias can help when the model doesn't represent a feature by taking on the expected value.

Interference is similar to before but ReLU means that negative interference, or interference where a negative bias pushes it below zero, is "free" in the 1-sparse case.

Figure 3: Loss on 1-Sparse Vectors

5

An important observation from the Loss function on the 1-sparse case is that the ReLU function filters out all negative interference. This relationship encourages feature benefit (expressing more features), but this time without the penalty of negative feature interference on the loss due to the ReLU activation function. This allows the model to learn superposition and represent more features than dimensions, while reducing the loss.

Result 3 is arguably the most important result. As features become sparser, superposition becomes a better strategy. Sparse features minimize interference between different features when they share dimensions in the activation space. If only a few features are active at a time, the likelihood of overlapping activations in shared dimensions is small, allowing the model to use these dimensions more efficiently and reap the benefits of expressing more features with low risk of feature interference.

On the other hand, with dense features, the model learns to represent an orthogonal basis of the most important m features where m is the number of dimensions of the layer. Dense features activate many dimensions simultaneously, so to avoid interference, the model learns a set of vectors that are mutually perpendicular.

In practice, most features are indeed sparse. For example, most inputs don't refer to cheesecake or prokaryotic organisms. These sparser features are "almost orthogonal" in the activation space, which enables the representation of more features, as demonstrated by the Johnson-Lindenstrauss Lemma. The key result of this experiment is that in the nonlinear model, there's a shift from monosemantic to polysemantic neurons as feature sparsity increases.

The nonlinear activation function ReLU is commonly used in neural networks, so it was deployed in the Anthropic experiment. It's important to note that other nonlinear activations, like a sigmoid function would also induce superposition in a similar fashion, but perhaps not to the same degree. Sigmoid networks tend to have more blurred and distributed representations since each neuron gets a degree of importance between 0 and 1. Then, many neurons slightly activate for most inputs, reducing sparsity, and hence polysemanticity.

4 The Multilayer Perceptron

Researchers speculate that the Multilayer Perceptron Layer in a transformer is a place where neurons are extremely polysemantic. To understand why, it helps to have context on how it functions.

At a *very high level*, a Transformer's architecture[6] is such that:

1. Input (e.g. text) is passed into the model.
2. This text is split into tokens, where one can think of each word roughly resembling a token.
3. Each token is embedded into a unique vector of high-dimension (e.g., 12,288) that represents the token in the high-dimensional space.
4. Each initial word embedding is updated by getting passed into an encoder block. Each encoder block contains two subcomponents:
   (a) A self-attention mechanism, which updates the components of the vector embedding, focusing on context of the input sequence (e.g. in

[6] Attention Is All You Need: `https://arxiv.org/pdf/1706.03762`

the input "river bank", the word river adds information to what sort of bank one is referring to).

(b) MLP. A feed-forward neural network which processes the output of the attention mechanism, still adding meaning to the word embedding.

5. A decoder block which takes in the embeddings from the encoder, and produces a set of probabilities of what the next token prediction should be. The decoder also uses an attention mechanism and MLP.

6. A final (text) output.

The role of the encoder is to add as much meaning as possible to the vector embedding of a word. After context from the input sequence is baked into the word embedding from the attention mechanism, the word embedding vector is passed through the MLP, a dense feed forward neural network to add additional meaning to the embedding. Mathematically, the MLP is usually a two-layer neural network that takes a word embedding vector:

$$\text{FFN}(\mathbf{x}) = \text{ReLU}(\mathbf{x}W_1 + b_1)W_2 + b_2$$

$\mathbf{W}_1$ and $\mathbf{W}_2$ are learned matrices during training. To understand why polysemanticity might transpire here, let's break down the MLP further.

$\mathbf{W}_1$ maps the 12,288-dimensional (exact dimension varies on the model) word embedding to a higher n-dimensional space. This n-dimensional space can be as high as 50,000[7] in practice (GPT-3).

$$\mathbf{x} \cdot W_1 = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{12287} \end{bmatrix} \cdot \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{12287,0} & w_{12287,1} & \cdots & w_{12287,n} \end{bmatrix}$$

$$= \begin{bmatrix} y_0 & y_1 & \cdots & y_n \end{bmatrix} \text{ where } \mathbf{W}_1 \in \mathbb{R}^{12,288 \times n}$$

Each component of $\mathbf{y}$, y_i, is a dot product of R_i, the i-th row of W_1, and $\mathbf{x}$:

$$\mathbf{y} = [R_0 \cdot \mathbf{x}, R_1 \cdot \mathbf{x}, \ldots, R_n \cdot \mathbf{x}]$$

In a perfectly interpretable world, one can think of each row of the matrix as asking some question about the word embedding to gain more information. For example, if $\mathbf{x}$ embeds the word "Einstein", and R_0 is related to basketball, the dot product $R_0 \cdot \mathbf{x}$ will have a near 0 or negative value, since the two are seemingly unrelated. However, if R_1 is related to physics, $R_1 \cdot \mathbf{x}$ will have a high dot product, and hence y_1 will be significant. This is because R_i and $\mathbf{x}$ are in a

[7]Parameter Counts: `https://benlevinstein.substack.com/p/a-conceptual-guide-to-transformers`

similar direction in space, and the direction of a vector indicates some meaning. A bias term is added to improve learning and help discard noise.

A nonlinear ReLU activation function is then applied to $\mathbf{y}$, setting all negative y_i to 0. As a reminder, each y_i can be thought of as a neuron, and after the ReLU, where each positive y_i indicates an active neuron and a 0 value means the neuron is unactive.

Then, a similar process is repeated, mapping $\mathbf{y}$ back down to the original dimension of its embedding (e.g. 12,288). The final result is to add this new vector to the original vector of same dimension that initially flowed into the MLP. This addition of vectors can be thought of as adding a new direction (or meaning) to the initial word embedding. For example, if the vector for Albert Einstein is passed into the MLP, the vector corresponding to relativity, might be added to the Albert Einstein vector, creating a richer vector. This example just describes the process for one embedding vector getting passed through the MLP, but in practice, all word embeddings from the input are passed in and processed in parallel.

However, it is rarely found that these individual neurons (e.g. y_i) represent a distinct feature. This is when one describes the neurons as behaving polysemantically, or exhibiting superposition. In practice, $\mathbf{y}$ might live in a 50,000 dimensional space, which is certainly a high enough dimension to where the network can leverage the Johnson-Lindenstrauss Lemma, or the ability to store exp(50,000) features. This means, a given feature would activate multiple y_i.

The MLP provides an intuitive structure for examining superposition. Not only does the MLP layer establish a privileged basis with the ReLU activation function, but it also differentiates between various interpretations of the same token and adds meaning to each. Input features are likely sparse, so high levels of polysemanticity will be likely.

5 Experiment

Based on the theoretical reasoning behind why the first layer of an MLP might exhibit superposition, I conducted an experiment to analyze neuron activations in this layer for various inputs. For this, I used BERT base-uncased[8], a pretrained transformer model from Hugging Face. BERT represents the encoder portion of the Transformer architecture, while GPT typically refers to the decoder portion of a Transformer model. BERT Base-Uncased embeds tokens into 768 dimensional space, performs self-attention on the embedding vector, and then passes the embedding into the MLP, where:

$$\text{FFN}(\vec{x}) = \text{GeLU}(\mathbf{x}W_1 + b_1)W_2 + b_2$$

$$\text{where } \mathbf{W}_1 \in \mathbb{R}^{768\times 3072}$$

This model is clearly much smaller than GPT-3, as it maps word embeddings into 3072 dimensions. The goal of this experiment is to investigate the polysemanticity of these 3072 neurons. Note that the GeLU activation function differs

[8]BERT Base-Uncased: `https://huggingface.co/google-bert/bert-base-uncased`

from ReLU in that it scales the input by the CDF of the input in a Gaussian distribution, but ultimately behaves similarly to ReLU in terms of inducing superposition.

I passed 3 text inputs into the model to try and control for noise from the input. The results were very similar across all inputs.

5.1 Text Inputs Used in the Model

I passed 4 inputs for each of the 3 distinct categories of text inputs:

1. Language

"The cat sat on the mat."
"She enjoys learning new languages."
"Die Katze sitzt auf der Matte."
"Sy hou van om nuwe tale te leer."

2. Math and Logic

"What is 25 plus 17? The answer is 42."
"A triangle with sides 3, 4, and 5 has a right angle. The area is 6."
"If Alice has 10 apples and gives 4 away, she has 6 apples left."
"The statement 'All men are mortal' is true. Socrates is a man."

3. Miscellaneous

"Water boils at 100 degrees Celsius under standard conditions."
"The sunset painted the sky in hues of orange and red."
"I love sunny days because they make me feel energetic and happy."
"Harry Potter defeated Voldemort in the Battle of Hogwarts."

On each of these inputs, I did an analysis on the activations of each of the 3072 neurons in the first layer of the MLP.

5.2 Cosine Similarity

In the perfectly interpretable world, each of these 3072 neurons corresponds to some distinct feature that resembles some direction in 3072-dimensional space. This would be the case if all of these vectors were perfectly orthogonal. Researchers suspect that neural networks leverage the "almost orthogonality" from the Johnson-Lindenstrauss Lemma that would allow the embedding space to represent exp(3072) features. To compute cosine similarity:

1. **Normalize each vector to have unit length:** Each activation vector $\mathbf{y}_i$ is normalized to have unit length by dividing it by its magnitude:

$$\hat{\mathbf{y}}_i = \frac{\mathbf{y}_i}{\|\mathbf{y}_i\|}$$

After normalization, all vectors have a magnitude of 1. This is done so that the magnitude of the vector doesn't affect neuron similarity, which is a direction in high dimensional space.

2. **Compute the cosine similarity matrix:** The cosine similarity between two vectors $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_j$ is computed as the dot product of the normalized vectors:

$$\text{cosine\_sim}(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) = \hat{\mathbf{y}}_i \cdot \hat{\mathbf{y}}_j$$

Since the vectors are unit vectors, the denominator is 1, so the cosine similarity is simply the dot product.

3. **Compute the cosine distance matrix:** The cosine distance is calculated as:

$$\text{cosine\_dist}(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) = 1 - \text{cosine\_sim}(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)$$

This is done for interpretability, where a value of 0 indicates identical vectors, a value of 1 indicates orthogonal vectors, and a value of 2 indicates vectors in opposite directions.

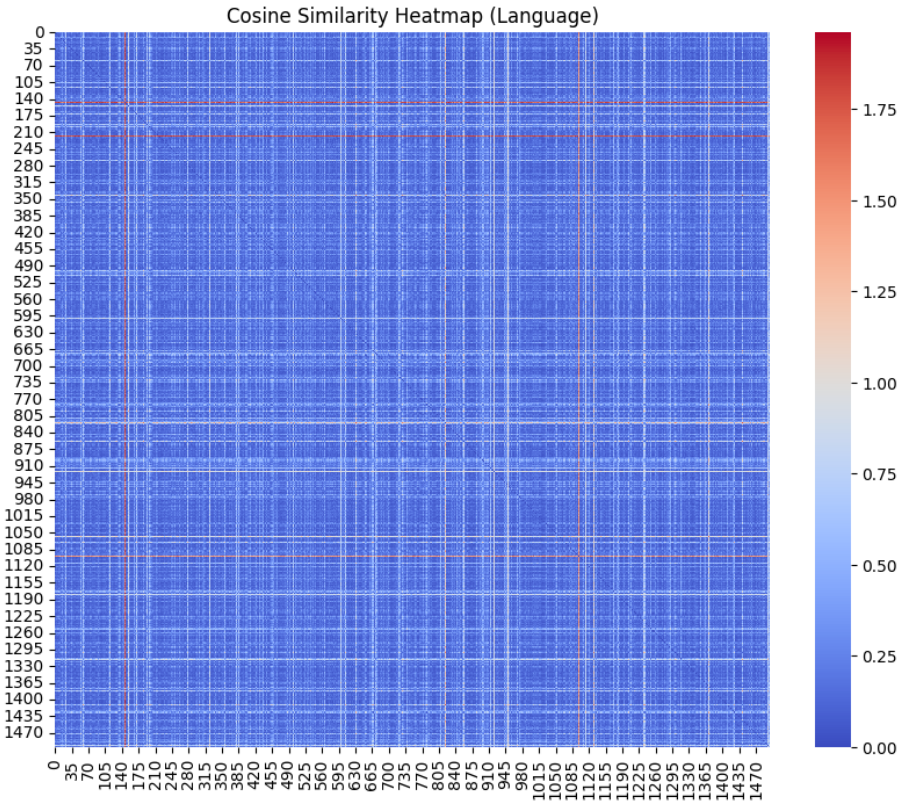

Figure 4: Cosine Similarity Heatmap

Figure 4 shows the heatmap used to visualize the normalized cosine distance matrix. Note that the similarities between only half of the neurons in the layer is shown for image clarity – the other half layer follows a similar pattern. One observation about this heatmap is that it has this plaid-like pattern. Most of the heatmap is blue, indicating near 0 cosine distance scores between neurons (i.e. their activation directions, or "meaning" is very similar), but there exist some white and red bands that stretch across the image.

The red bands in the heatmap, though infrequent, represent a cosine distance of 2, indicating that the vectors are nearly opposite in direction. Some neurons consistently show a cosine distance close to 2 with every other neuron in the network, suggesting that these neurons may be specialized to detect distinct, contradictory, or non-overlapping features in the data.

In contrast, the more frequent white bands in the heatmap indicate neurons that are orthogonal to all others, meaning they have a cosine distance of 1. This is an interesting result, as it suggests that there exist neurons that are individually dedicated to representing unique features, as their activations are uncorrelated with those of other neurons in the network.

On the whole, the heatmap is majority blue, indicating that the neurons are highly correlated with each other. If each neuron truly represented a unique feature, then the heatmap would be white (value 1) as all vectors would be orthogonal. As theorized above, this orthogonality is present in linear models, but the GeLU and sparsity of inputs in the MLP contributes to the polysemanticity present.

This high rate of polysemanticity is further affirmed by *Figure 5*, showing a left-skewed distribution of cosine distance values.

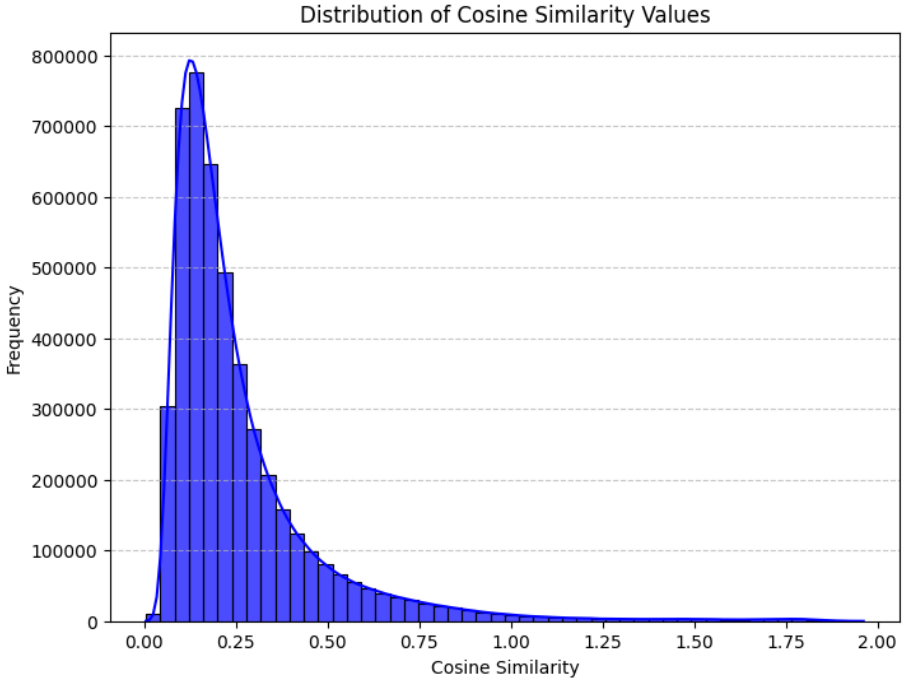

Figure 5: Cosine Similarity Histogram

These trends are consistent across all 3 text inputs[9]

5.3 Distributions of Neuron Activations

When looking more individualistically at the neurons, it's clear that superposition is present in the layer. I sampled 50 random neurons and plotted their activation value corresponding to each token input, examining to see if many neurons are active for the same token.

[9]Colab Results: insert github

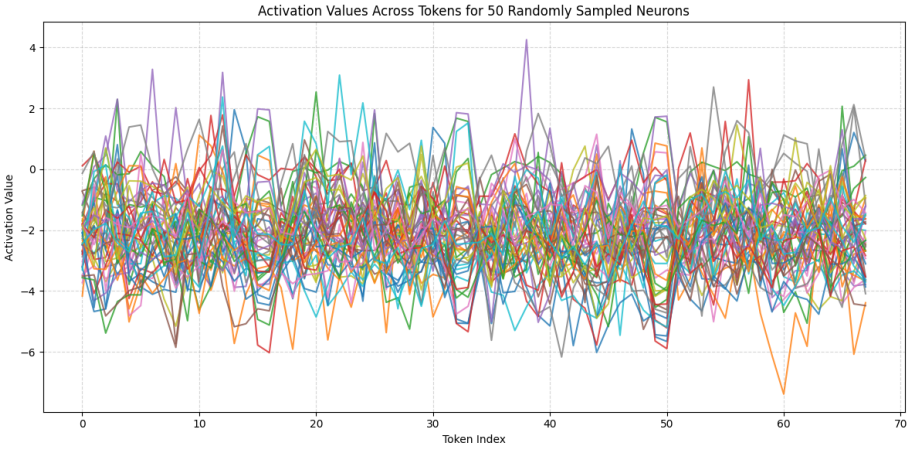

Figure 6: Distribution of Activations for Randomly Sampled Neurons

In *Figure 6*, there clearly exist high activation values for many neurons for each token in the input. This trend is consistent across random samples and text inputs. If the neurons did not exhibit superposition, we might expect only one neuron to be active while the rest hover around 0 for each token. The amount of noise present in *Figure 6* suggests otherwise. *Note*: Disregard the consistent activation values across the neurons sampled around tokens 16, 33, and 49. These tokens are "padding" tokens that have no meaning and are only used for computational purposes.

To get a better sense of how neurons were being activated across the network, I summed all activation values for each neuron firing in response to all the tokens from the 3 different text inputs. Here's how the histogram looked:

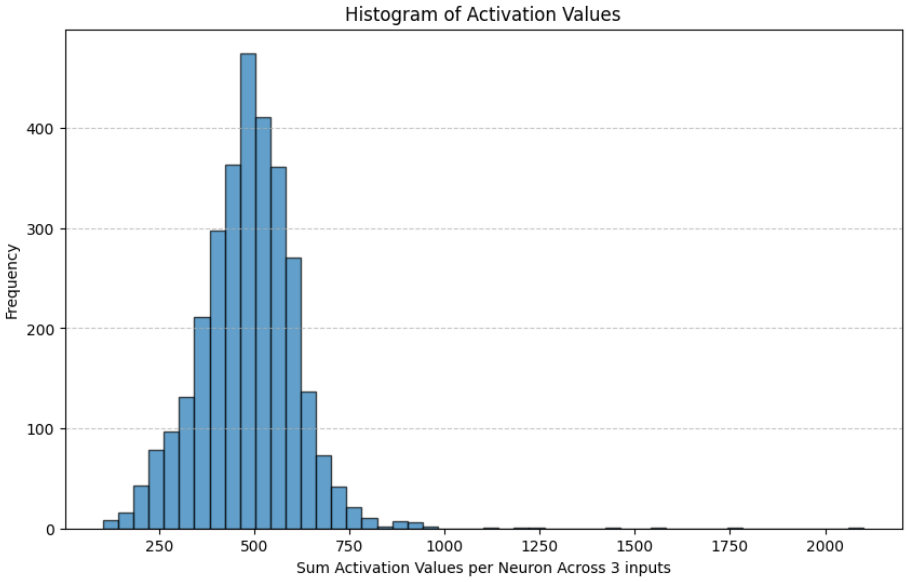

Figure 7: Activation Values per Neuron

This histogram shows a normal distribution of activation values across neurons centered at activation value 500. The fact that these summed neuron activations follow a normal distribution suggests that the network does not rely on a single neuron for a particular feature but rather distributes the responsibility of different features across neurons. Neurons might not be specialized to only one concept, but rather may represent multiple overlapping concepts or meanings.

6 Conclusion

The sparsity of features and the nonlinearity often employed for denoising in neural networks contribute to learning dynamics that encourage the superposition of neurons. This capability allows neural networks to theoretically represent exponentially more features than the number of dimensions in the network, making them incredibly powerful tools. However, this same property introduces challenges in understanding individual neuron behavior, which is critical for advancing AI alignment.

In this study, I analyzed the activation patterns of neurons in the first layer of BERT's MLP, focusing on their polysemanticity and capacity for superposition. By examining cosine similarity and activation distributions, I found that most neurons are not specialized for representing distinct, non-overlapping features. Instead, the activations suggest that neurons participate in encoding

multiple features, highlighting the high degree of polysemanticity characteristic of modern transformer-based models.

References

[1] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, and Martin Wattenberg, *Toy Models of Superposition*, 2022, https://transformer-circuits.pub/2022/toy_model/index.html.

[2] Grant Sanderson, *How Might LLMs Store Facts?*, 2024, https://youtu.be/9-Jl0dxWQs8?si=5BOu26gNvM0FHt1i.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient Estimations of Word Representations in Vector Space*, 2013, *arXiv preprint arXiv:1301.3781*, https://arxiv.org/abs/1301.3781.

[4] William Johnson and Joram Lindenstrauss, *Johnson-Lindenstrauss Lemma*, 1984, https://en.wikipedia.org/wiki/Johnson%E2%80%93Lindenstrauss_lemma.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is All You Need*, Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008, 2017, https://arxiv.org/abs/1706.03762.

[6] Ben Levinstein, *A Conceptual Guide to Transformers: Part I*, 2023, https://benlevinstein.substack.com/p/a-conceptual-guide-to-transformers.