20011262

INST0062 MSc Dissertation

Luke Dickens

This dissertation is submitted in partial fulfilment of the requirements for the Master's degree in Knowledge, Information and Data Science, UCL

word count = 6748

Can Sparse Autoencoders disentangle superposed features?

September 13, 2025

1 Abstract

Abstract

Neural networks often exhibit superposition, where multiple features share representational space, producing polysemantic neurons that hinder interpretability. Prior work (Toy Models of Superposition) demonstrated that superposition emerges in a critical sparsity region, while Towards Monosemanticity proposed sparse autoencoders (SAEs) to recover monosemantic features. However, recent critiques suggest that SAEs may instead split, duplicate, or rotate features, raising doubts about their reliability.

In this paper, we reproduce the toy model setup to analyse superposition systematically. We examine weight matrices and low-dimensional geometries to track phase transitions in feature encoding, and extend the analysis using representational similarity metrics (RSA, linear CKA, RBF-CKA, and Procrustes). We further test whether SAEs of mirrored dimensionality align with original autoencoder features.

Our results show that superposition emerges stochastically within the critical region, producing qualitatively distinct geometries with no consistent pattern across runs. While all similarity metrics are broadly consistent, Procrustes analysis is uniquely sensitive to representational shifts, making it a strong candidate for future work. We conclude that while SAEs can reconstruct inputs, their hidden representations often diverge, highlighting limitations in their ability to faithfully decompose superposed features.

2 Introduction

Deep neural networks achieve impressive capabilities across domains, however the computations they perform arise from stochastic learning and so remain poorly understood. A better understanding could illuminate paths for improving their performance and would make safer their widespread usage. This has motivated Mechanistic Interpretability, a research field aiming to connect network's internal structure to their downstream behaviour. A key obstacle is

polysemanticity, where single neurons encode multiple unrelated features simultaneously [17]. This complicates understanding them, since it is unclear whether the structures we identify correspond to real features or artefacts of training.

Recent work has proposed that polysemanticity arises from the phenomenon of "superposition", in which sparse, high-dimensional features can be faithfully stored in lower dimensional representations, where features share dimensions. Toy Models of Superposition [5] demonstrated that when autoencoders compress sparse synthetic inputs, features overlap producing a phase transition in representational geometry around a critical sparsity region. Building on this, Towards Monosemanticity [4] introduced sparse autoencoders (SAEs) to expand hidden dimensions in an attempt to disentangle superposed features. However, subsequent critiques [3, 15, 13] argue that SAEs may suffer from instability, feature splitting, or simply rotate the same subspace without revealing genuinely new structure.

In this paper we revisit these foundations. We reproduce the setup of Toy Models to analyse the emergence of superposition. We then test whether SAEs of mirrored dimensionality recover the original autoencoder features, or whether their hidden representations diverge through analysis with alignment metrics. Our contributions are threefold:

1. We visualise superposition in both weight matrices and feature geometries across the critical sparsity region.

2. We evaluate alignment between autoencoders and SAEs using four complementary metrics: RSA, linear CKA, RBF-CKA, and Procrustes analysis.

3. We provide evidence that superposition is highly stochastic and that Procrustes alignment is particularly sensitive to the phase change in representations.

3 Background

3.1 Mechanistic Interpretability

Neural networks empirically learn a function by gradually adjusting their weights to improve performance on training samples through gradient-based optimization. Good performance on held-out data is evidence that the network is able to extract key information that is useful for predicting the expected output [12, 23]. Early mechanistic interpretability work aimed to discover what neural networks had learned by analysing the smallest unit they can be decomposed to, neurons. Olah et al. introduced the concept of "circuits", which are discovered by passing a series of related inputs through a network and identifying which neurons activate most strongly, i.e. which units are used to reach the output [18]. It was theorized that by breaking a neural network down into a collection of circuits, the overall function could be decomposed into bespoke subfunctions that humans could interpret. However, scaling this approach to larger models

proved ineffective, with later studies finding that single tasks could not be consistently ascribed to distinct sets of neurons [19]. There is no strong motivation for networks to cleanly separate features to individual neurons, and so techniques that analyse existing structure often find that processes are distributed across the whole network, making them too complex for people to decipher [8].

3.2 Superposition

This has given rise to the theory of superposition, the idea that a single neuron can activate on and encode multiple features at once [5, 14]. This phenomenon is especially relevant in large models like LLMs, where the number of features they can recognise vastly exceeds the dimensionality of their hidden layers [5, 14]. Early doubts surrounding the effectiveness of neural networks centred on the apparent intractability of representing functions over extremely high-dimensional input spaces. Researchers have argued that this is tractable because real-world data is highly constrained in its distribution [1, 6]. For example the input space of a vision model consists of all possible arrangements of pixels within a certain height, width and spectrum of colours. However in practice, most of the possible images will correspond to meaningless noise and if our task was classifying images of animals, much of the image (e.g., corners and edges representing background texture) is irrelevant to the task, meaning the domain of the function our network aims to learn is constrained such that the complex structure of inputs lies close to a lower-dimensional manifold, which can often be approximated locally by a linear basis in hidden space [1]. The hidden neurons can be understood as analogous to a set of basis vectors that span this manifold, with the network encouraging diversity of directions, though not strict linear independence. Thus superposition enables networks to represent more features than the hidden layer has dimensions [5]. This view is supported by the discovery in toy models, that properties like sparsity and importance are necessary for superposition to occur [5]. In their experiment they use autoencoders, a self-supervised learning technique that aims to reconstruct inputs after passing them through one hidden layer with a smaller dimension than the input. This technique was originally intended to compress data for more efficient storage, but anthropic showed that under the right conditions it can force features into superposition.

3.3 Disentangled Features

Superposition arises naturally from the training process so constraining the behaviour of a model during training can coerce it to encode neurons differently. To address this, researchers have explored constraints that encourage networks to learn cleanly separable features. However in the case of large models, like LLMs, neurons are often polysemantic, activating on multiple unrelated features simultaneously [14], and applying constraints without degrading performance is difficult. Early work, such as Ng's sparse autoencoders [16], introduced a sparsity penalty on its hidden layer's activations to reduce the neurons average ac-

tivation rate across the dataset, yielding a sparser representation of the inputs. More recently, Anthropic's Towards Monosemanticity [**elhage2023**] applied a per-input L1 sparsity penalty, so that each example activated 1-2 neurons. By using a hidden layer of much greater size than their input, they aimed to decompose existing superposed features from the hidden layers of LLMS, into a larger set of monosemantic neurons. By grouping inputs according to which neuron fires, they could identify common patterns where each neuron ideally corresponds to a distinct, interpretable feature, turning them into a feature dictionary.

This approach has faced several critiques. Single coherent features that should be one feature can be divided across multiple neurons ("feature splitting") [4, 15]. The feature dictionaries learned are not unique because they arise through a stochastic learning process [17]. Part of the reason for this is that we have no knowledge of how many features are superposed in the network's low-dimensional representations, so the choice of the SAE's hidden layer dimension is arbitrary [5]. The L1 penalty biases the learning to divide features evenly across neurons, which can lead to over-fragmentation if the hidden layer size is too great or under-fragmentation if it is too small [4, 2]. Additionally, without being able to replicate the same dictionary across runs, we cannot know whether the dictionary faithfully reflects the network's internal representations or is an artifact of the method [15].

The Toy Models of Superposition paper provides strong evidence for the existence of superposition, but also highlights that the phenomenon is not yet fully understood [5]. The approach in Towards Monosemanticity relies on the assumption that the SAE's expansion into a higher-dimensional space extracts features in a similar way to how an autoencoder reconstructs inputs from a superposed representation [4]. The autoencoder is directly encouraged to recover the original features in its final layer by its loss function, but this is not the case for the hidden layer of the SAE, which could use any representation that still permits reconstruction [17, 2].

3.4 RSA

Formally we are discussing the comparison of two functions that map m-dimensional inputs to n-dimensions where $m < n$. Since these functions are described by the weight matrix (and bias vector) of the corresponding layer it would be reasonable to use these parameters as a starting point. However, different parametrizations can perform the same function, for example if one model's hidden layer is a rotation of another's. Additionally this weight matrix is just one component of the overall network function. Scaling up the first matrix and inverse-scaling the second would leave the result unchanged, so matrices performing the same task could look vastly different. Some of these challenges are addressed in Toy Models of Superposition [5], where a correlation matrix $W^\top W$ is used to analyse feature activations. By comparing the columns' overlap with each other, rotation invariant structure in the weight matrix is revealed and used to demonstrate phase change in the encoding across the critical sparsity region where superpo-

sition occurs. $W^\top W$ captures the overlap structure between features ignoring arbitrary basis choices. Diagonal structure suggests orthogonally encoded features while large terms elsewhere indicate overlap in direction and therefore superposition.

Another popular approach is to view these functions in terms of where they map the same input space. Representational Similarity Alignment (RSA) metrics do this by comparing Representational Dissimilarity Matrices (RDMs). Given a representation matrix $X \in \mathbb{R}^{n \times d}$, where n is the dimensionality of the hidden layer and d the number of datapoints, the Representational Dissimilarity Matrix (RDM) is defined as

$$\text{RDM}_{ij} = \|x_i - x_j\|^2,$$

where $x_i, x_j \in \mathbb{R}^n$ are the representations of datapoints i and j. A distance metric is applied pairwise across the d datapoints, measuring how similarly the same inputs have been encoded in relation to each other in the two subspaces. The motivation behind this is that elements will be arranged with a similar structure in both subspaces regardless of whether they differ in terms of scale or rotation [11, 10].

4 Research Questions

The research questions this paper aims to answer are as follows:

1. **What conditions give rise to superposition in neural networks?** Superposition has been characterised as a phase change in model behaviour. By focusing on the critical sparsity region identified in Toy Models of Superposition [5], we aim to clarify the factors that lead to its emergence.

   Hypothesis: We expect superposition to emerge only within a narrow band of sparsity values, with models outside this region exhibiting either cleanly separated features or failing to accurately reconstruct representations.

2. **How does the geometry of representation space change across this phase transition?** Visualising low dimensional projections provides insight into how features are "pushed together" as superposition occurs.

   Hypothesis: We expect to observe a qualitative shift in geometry, indicative of the occurence of a phase change.

3. **How does representational alignment between autoencoders and sparse autoencoders vary with superposition?** We measure alignment between the AE's features and those decomposed by the SAE using Representational Similarity Analysis (RSA) metrics, in order to test whether the SAE faithfully captures the same underlying structure.

   Hypothesis: We expect representational alignment to remain high outside the critical region but to decrease significantly within it, reflecting a different encoding of features, arrived at through superposition.

5 Methodology

First leg

We begin by reproducing the experiment in Anthropic's Toy Models of Superposition [5]. as the foundation of our method. The motivation for this is three-fold. First, it is a prominent work in mechanistic interpretability and therefore by building on it our work will be more accessible to others in the field. Second, the concepts involved, such as superposition, are relatively new and yet to be fully understood. Additionally, subsequent works in mechanistic interpretability have raised concerns that sparse autoencoders may learn artefacts unrelated to the true underlying features, rather than reliably extracting features [15]. Reproducing the setup from this paper allows us to give more credence to our results being the product of superposition rather than artefacts of the implementation. Third, the aim of this experiment is to critique the conclusions found in Toy Models. Producing evidence in their setup allows us to draw a connection more easily.

We train autoencoders of varying dimension (expressed as $n{-}m{-}n$, where n is the dimension of the input vectors and reconstructions and m is the dimension of the hidden layer) on input data of varying sparsity and importance.

5.1 Architectures

The $20-5-20$ architecture was chosen to recreate the investigation from Toy Models of Superposition [5]. The $10-5-10$ and $30-5-30$ architectures were chosen to investigate the impact of decreasing and increasing the difference between input/reconstruction and hidden dimensions, respectively. Toy Models also demonstrated that superposed features exhibit different geometric structures depending on these dimensional ratios.

Additionally, we have run experiments with smaller architectures ($3-2-3$, $5-2-5$, $4-3-4$, and $8-3-8$) to test whether we can recover the superposition geometries discovered in Toy Models, and for ease of visualisation, as the resulting polytopes can be plotted directly in two or three dimensions.

5.2 Activation Function

We use the Rectified Linear Unit (ReLU),

$$\mathrm{ReLU}(x) = \max(0, x),$$

because it is standard in neural network architectures due to its simplicity and ability to mitigate vanishing gradients.

5.3 Loss Function

We minimise a weighted reconstruction error,

$$\mathcal{L}(x, \hat{x}) = \sum_{i=1}^{n} I_i \left(x_i - \hat{x}_i\right)^2,$$

where the importance weights are defined elementwise by

$$I_i = \alpha^{\,i-1}, \qquad \alpha = 0.7.$$

Just like sparsity, real-world features also vary in importance; for example, a positive/negative sentiment dimension in a word embedding may have greater impact on downstream behaviour than a gender dimension. We artificially control this using an exponentially decaying importance vector $I = (\alpha^0, \alpha^1, \ldots, \alpha^{n-1})$, as was done in Toy models [5]. Although varied importance is not necessary for superposition to occur, it has interesting impacts on the result, for example prioritising a dedicated dimension for more important features and compressing less important ones onto the same dimension.

5.4 Data Generation

Inputs are generated as

$$x_{ij} = B_{ij}\, U_{ij},$$

with

$$B_{ij} \sim \text{Bernoulli}(1 - S), \qquad U_{ij} \sim \text{Uniform}(0, 1),$$

where $S \in [0, 1]$ is the sparsity parameter controlling the probability a feature is inactive. Following [5], we vary S across $[0, 1]$ with a focus on the critical region $(0.7 < S < 0.9)$.

5.5 Optimiser

All models are trained using the Adam optimiser [9] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Parameters are updated as

$$\theta_{t+1} = \theta_t - \eta\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

5.6 Analysis Functions

To evaluate the autoencoders trained in the first leg, we compute diagnostic measures that capture feature interactions and superposition. As in Toy Models[5], the superposition matrix (and accompanying bias vector) is defined as

$$M = W^T W, B = b$$

where $M \in \mathbb{R}^{n \times n}$ measures the extent to which different features interfere in the learned representation.

Second leg: Sparse Autoencoders

While the first leg of our methodology focuses on training and analysing superposition induced in standard autoencoders, the second leg investigates whether Sparse Autoencoders (SAEs) can be used to extract features out of superposition. This extension is directly motivated by Anthropic's work [4], where SAEs were proposed as a means of decomposing polysemantic representations into more interpretable monosemantic features.

5.7 Sparse Autoencoders (SAEs)

Formally, given an input $x \in \mathbb{R}^m$, the encoder produces sparse hidden activations

$$h = \mathrm{ReLU}(W_{\mathrm{enc}} x + b_{\mathrm{enc}}),$$

which are then decoded to reconstruct the input

$$\hat{z} = W_{\mathrm{dec}} h + b_{\mathrm{dec}}.$$

The loss combines mean squared reconstruction error with an ℓ_1 sparsity penalty on hidden activations:

$$\mathcal{L}_{\mathrm{SAE}} = \frac{1}{N} \sum_{i=1}^{N} \|z_i - \hat{z}_i\|^2 + \lambda \frac{1}{N} \sum_{i=1}^{N} \|h_i\|_1,$$

where λ controls the strength of the sparsity constraint.

Several critiques have been raised of this approach. These include feature duplication or splitting across multiple neurons [3], instability of features across runs [15], and the possibility that SAEs simply provide a rotated reparameterisation of the same subspace rather than uncovering genuinely new features [13]. A key assumption in Towards Monosemanticity [4] is that projecting activations into a higher-dimensional sparse code is analogous to the compression–decompression process in standard autoencoders.

To colarify this, we invert the dimensions: for an autoencoder trained with architecture $n-m-n$, we train a corresponding sparse autoencoder with dimensions $m-n-m$ on its bottleneck activations. This ensures that the SAE's first projection takes the same input space as the AE's bottleneck, enabling direct comparison of the two representations. We evaluate this relationship using representational similarity metrics (RSA, CKA, and Procrustes alignment).

5.8 Representational Alignment Metrics

To compare autoencoder (AE) and sparse autoencoder (SAE) representations, we employ four alignment metrics commonly used in representational similarity analysis. Each captures a complementary aspect of similarity.

Representational Similarity Analysis (RSA). RSA [11] compares two representational dissimilarity matrices (RDMs). Formally, for two representation sets $R_1, R_2 \in \mathbb{R}^{N \times d}$:

$$d_1^{(ij)} = \|r_1^{(i)} - r_1^{(j)}\|_2, \quad d_2^{(ij)} = \|r_2^{(i)} - r_2^{(j)}\|_2,$$

$$\mathrm{RSA}(R_1, R_2) = \mathrm{Corr}(d_1, d_2).$$

Rationale: RSA is invariant to rotation and scaling, and measures whether pairwise relationships between datapoints are preserved across spaces.

Centered Kernel Alignment (CKA). CKA [10] measures similarity between representation matrices using kernel-based dependence. Given $X \in \mathbb{R}^{N \times D_1}$, $Y \in \mathbb{R}^{N \times D_2}$, and centered Gram matrices $\tilde{K}$, $\tilde{L}$:

$$\mathrm{CKA}(X, Y) = \frac{\mathrm{HSIC}(X, Y)}{\sqrt{\mathrm{HSIC}(X, X) \cdot \mathrm{HSIC}(Y, Y)}},$$

where HSIC is the Hilbert–Schmidt Independence Criterion. Rationale: CKA is invariant to isotropic scaling and orthogonal transformations, making it robust for comparing learned feature spaces of different dimensionality.

RBF-CKA. We additionally compute CKA with a radial basis function (RBF) kernel [10]. The kernel is defined as

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right).$$

Rationale: Whereas linear CKA is sensitive to dot-product structure, RBF-CKA captures nonlinear similarity, complementing the linear case.

Procrustes Alignment. Orthogonal Procrustes analysis [21] finds the optimal rotation/reflection aligning two representation matrices:

$$R^* = \arg\min_{R \in O(d)} \|XR - Y\|_F,$$

with resulting similarity

$$\mathrm{Proc}(X, Y) = \frac{\mathrm{tr}(R^{*T} X^T Y)}{\|X\|_F \, \|Y\|_F}.$$

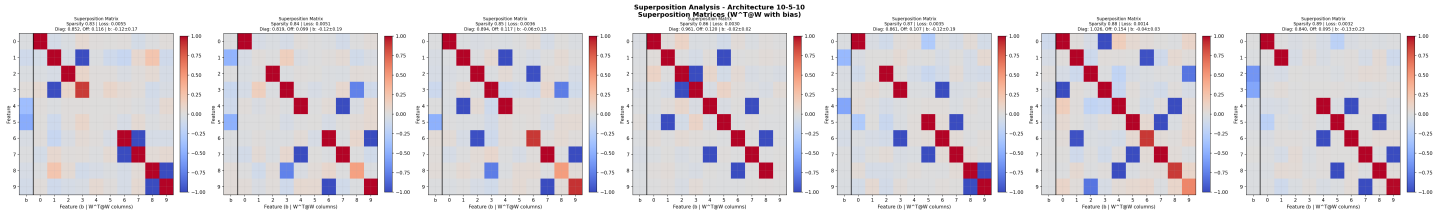
Rationale: Procrustes directly tests whether two spaces can be aligned by an orthogonal transformation, complementing kernel-invariant metrics.

Together, these metrics provide complementary perspectives on representational similarity: RSA captures relational geometry, CKA and RBF-CKA are invariant to to scaling/rotation (linear and nonlinear), and Procrustes tests rotational alignment. Using all four reduces the risk of a conclusions being the result of an assumption of one metric.

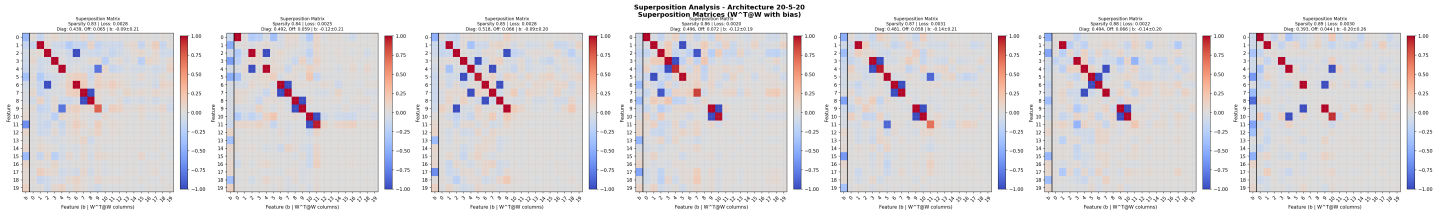
6 Evaluation

6.1 Superposition matrices at critical sparsities

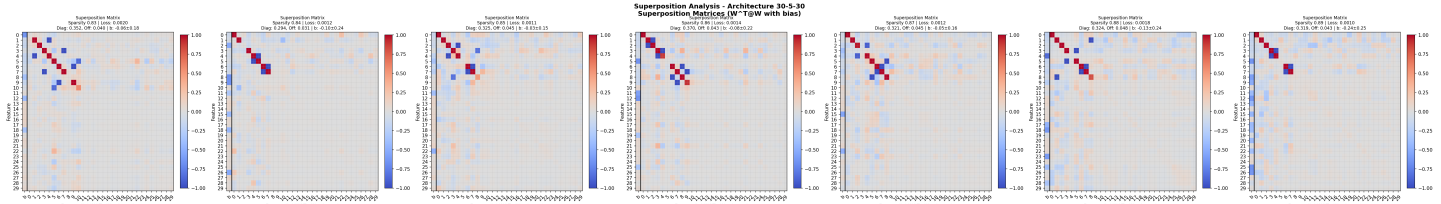
To illustrate the phase transition in feature representations, we show superposition matrices around the critical sparsity region for three architectures: 10-5-10, 20-5-20, and 30-5-30.



(a) 10–5–10 architecture



(b) 20–5–20 architecture



(c) 30–5–30 architecture

Figure 1: Superposition matrices across sparsity values for three architectures: (a) 10–5–10, (b) 20–5–20, and (c) 30–5–30.

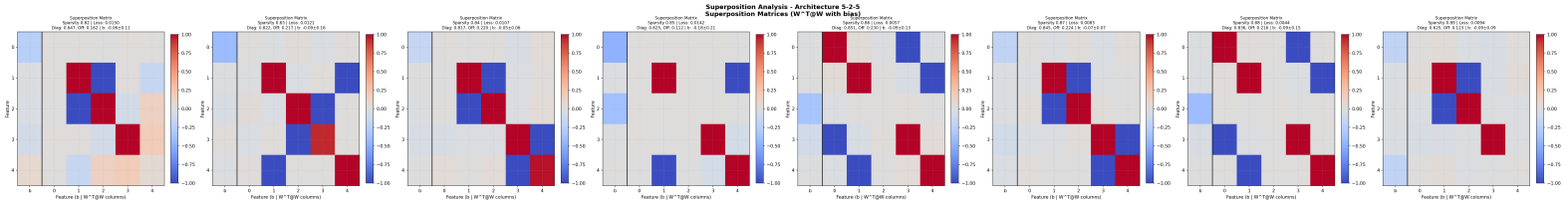
To examine how superposition arises, we plot the superposition matrices ($W^\top W$) across sparsity levels for each architecture. Figure 1 demonstrates a consistent phase change through this region across architectures. At sparsity 0.84 for size 20 and 30, and 0.83 for size 10, the most important features are captured by their own dedicated dimensions, either with a dominant weight or bias, and kept as orthogonal to each other as possible. The less important features are encoded in opposing directions within a shared dimension, described in Toy Models of Superposition as "antipodal pairs" [5]. These features interfere significantly, whilst minimising interference with the more important features. As sparsity increases the interference begins to distribute more evenly across the network, evidenced in the plot by more frequent blue squares, further from the diagonal. This behaviour subsides towards the end of the critical region as the plots begin to resemble those at the start of the region. To better understand this, next we shrink our hidden dimension size to directly visualise the geometry of features.

6.2 Feature Geometry

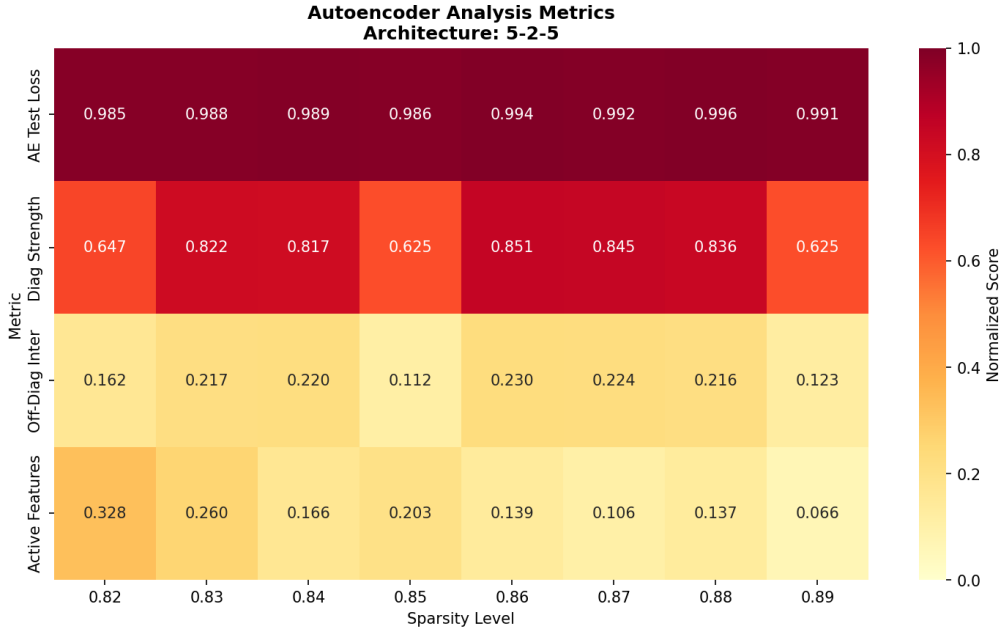
at lower dimensions, Figure 2(a) tells us that there is more instability in which encoding strategy is used, evidenced by qualitative differences between matrices at neighbouring sparsity values. In figure 2(b) The AE loss is consistently low, meaning the network is able to successfully recover the features in all cases. However looking at the feature geometry plots (Figure 3) tells us that the compressed representation it uses to arrive at this reconstruction can vary substantially.

Higher diagonal strength and off-diagonal interference are indicative of more features seeing strong representation in the encoding. This is generally observed when the geometry produces 4-sided polygons. Notably there is not one arrangement that produces this pattern, instead three different matrices correspond to the same geometry. Meanwhile lower diagonal strength and weaker interference show that less important features are neglected by the representation and instead are encoded by the bias vector, which activates more strongly in these cases. This explains the triangle shape for 5 features in figure 3(c), where the orthogonality of one feature is maximised in relation to two others, an antipodal pair, and the last two are captured by the bias vector.

This pattern is reinforced by the three-dimensional visualisations. Highest diagonal strength and off diagonal interference (at $s = 0.84$) corresponds to figure 5(b), a clear depiction of the square anti-prism described in Toy Models [5]. This case represents the strongest form of superposition, where interference is distributed as evenly as possible between features and the geometry approximates a regular polyhedron. In contrast, at $s = 0.86$, strong bias activations, weak diagonal strength and lack of interference suggest no features are superposed. The 3D plot shows a clear difference in shape from the previous case, lacking regularity. Three features activate strongly, orthogonally to each other and the rest are weakly represented.

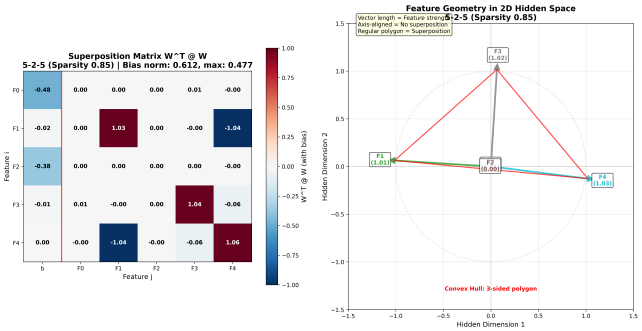


(a) Superposition matrices across sparsity values for 5–2–5 (seed 629).

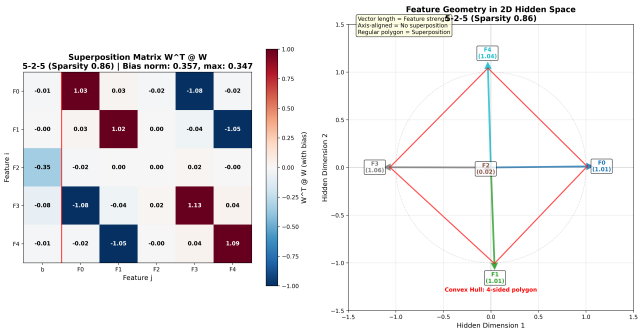


(b) Metrics heatmap for 5–2–5 (seed 629).

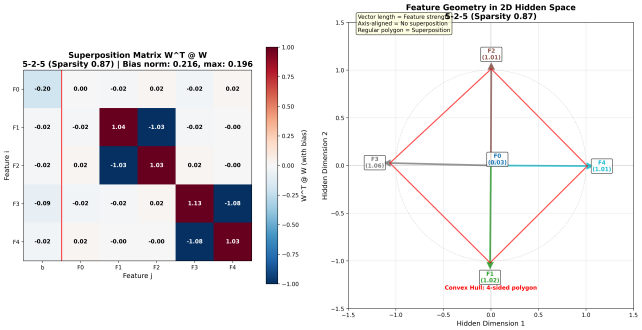
Figure 2: Comparison of superposition matrices (top) and metrics heatmap (bottom) for the 5–2–5 architecture, seed 629.



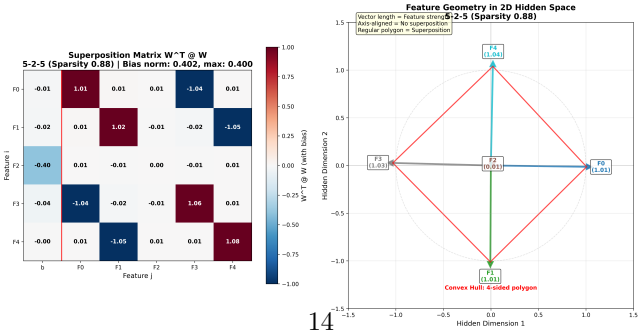
(a) 2D geometry for 5–2–5 at sparsity 0.85 (seed 629).



(b) 2D geometry for 5–2–5 at sparsity 0.86 (seed 629).

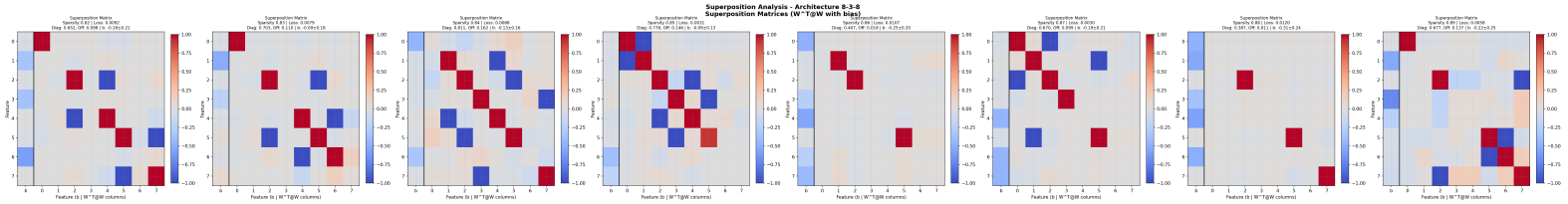


(c) 2D geometry for 5–2–5 at sparsity 0.87 (seed 629).

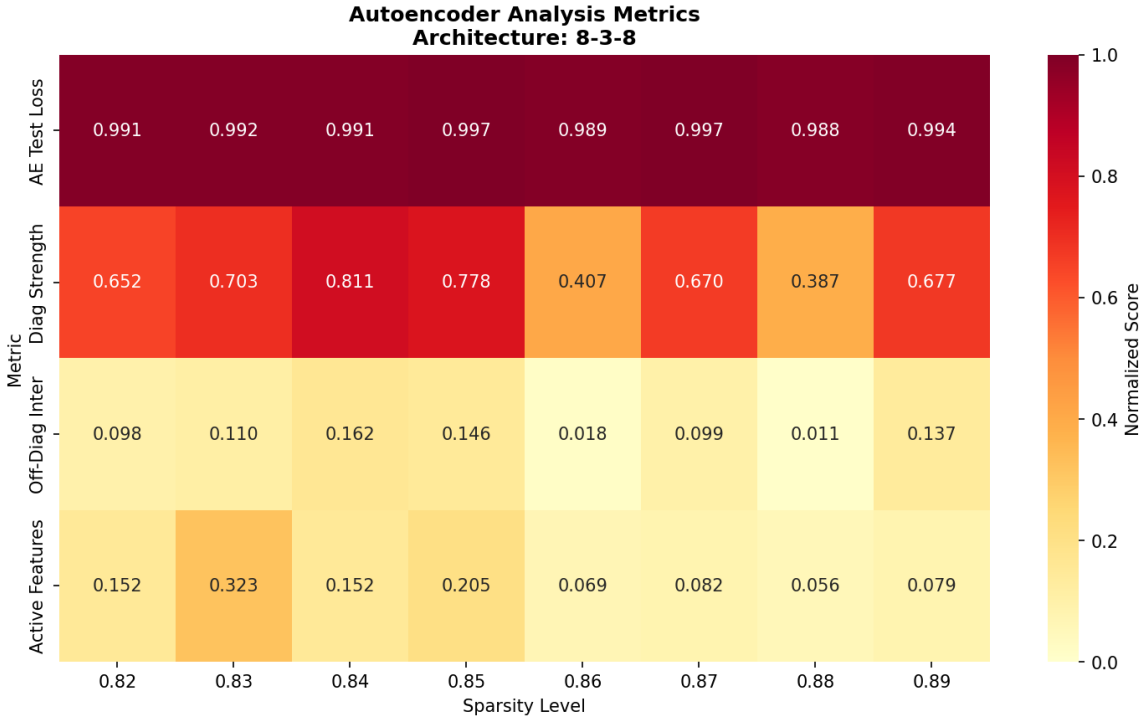


(d) 2D geometry for 5–2–5 at sparsity 0.88 (seed 629).

Figure 3: 2D geometry visualisations for the 5–2–5 architecture (seed 629) across four sparsity values: 0.83, 0.84, 0.85, 0.86.

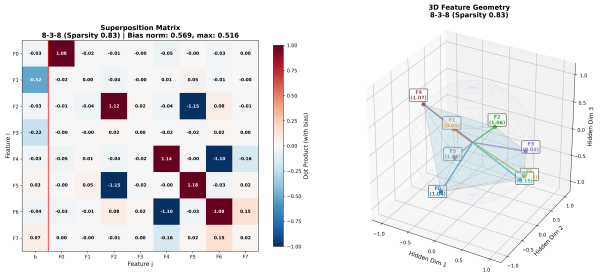


(a) Superposition matrices across sparsity values for 8–3–8 (seed 629).

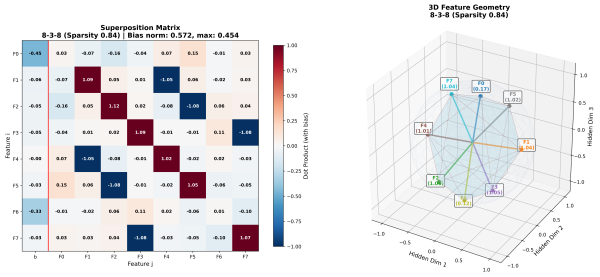


(b) Metrics heatmap for 8–3–8 (seed 629).

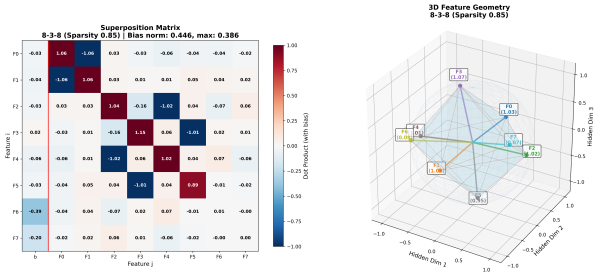
Figure 4: Comparison of superposition matrices (top) and metrics heatmap (bottom) for the 8–3–8 architecture, seed 629.



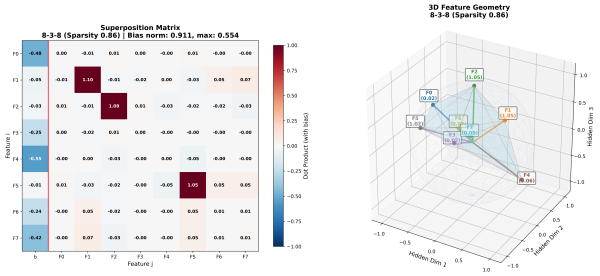
(a) 2D geometry for 8-3-8 at sparsity 0.83 (seed 629).



(b) 2D geometry for 8-3-8 at sparsity 0.84 (seed 629).



(c) 2D geometry for 8-3-8 at sparsity 0.85 (seed 629).



(d) 2D geometry for 8-3-8 at sparsity 0.86 (seed 629).

Figure 5: 3D geometry visualisations for the 8-3-8 architecture (seed 629) across four sparsity values: 0.83, 0.84, 0.85, 0.86.

6.3 RSA metrics

Returning to the higher dimensional architectures (figures 6, 7 and 8) we look to our alignment metrics to investigate the phase change more systematically.

The first noteworthy observation is that AE reconstruction loss and SAE reconstruction error remain consistently low across all architectures and sparsities. This confirms that every run is successfully reconstructing its inputs, and that variation in similarity metrics cannot be explained by model collapse.

In figure 8 (30 dimensional), Procrustes is the only metric showing strong signs of variance across our critical region, which suggests this is the most useful for capturing the phase change that is taking place.

At lower dimensions we see the trend fluctuate across the critical region. For example in figure 7 (20 dimensional), a sparsity of 0.85 achieves the highest similarity across all metrics whilst its neighbours are some of the lowest. This is the clearest evidence of a phase change occurring, as there must be a significant difference between the encoding strategy used at 0.84 and 0.85.

In figure 6 (10 dimensional), all alignment metrics fluctuate sharply across the region. This suggests that in low-dimensional settings, encoding strategy choice is fragile, as neighbouring sparsities produce vastly different scores.

One might expect RSA scores to decline towards the middle of the critical region and then recover at the end but our results suggest no such pattern exists. Instead the phase change is a region of instability between models, however metrics are internally consistent. there is a clear correlation between the performance of alignment. This is most visible in figure 6 where Both CKA and RSA scores congregate around a similar value for each training run and the Procrustes score is consistently ~ 0.45 below them in every case. In addition variance across these metrics is consistent for individual architecture-sparsity combinations. Variance seems to spike most at 0.85-0.86, suggesting this is the region where different runs are most likely to diverge in their encoding strategies.

Across all architectures, diagonal strength and off-diagonal interference vary randomly across sparsities with no apparent connection to any of our RSA metrics.

Finally, to validate that these trends are not the result of sparsity penalties forcing certain encodings, we performed a $\lambda \to L_0$ sweep (Appendix B). This confirmed that varying the penalty predictably controls the mean number of active neurons. However, it does not correlate with the divergence in alignment metrics, showing that L_0 is not influencing the nature of superposition in the model.

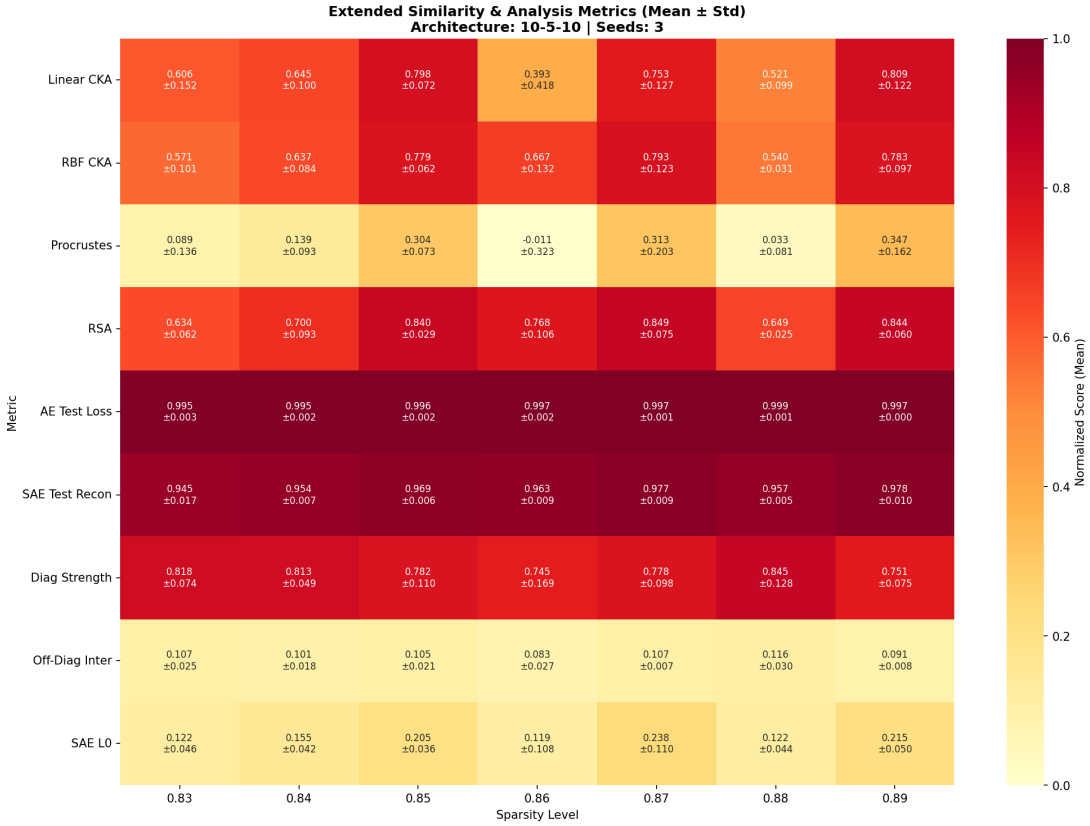


Figure 6: Extended similarity and analysis metrics across sparsities for the 10–5–10 architecture.

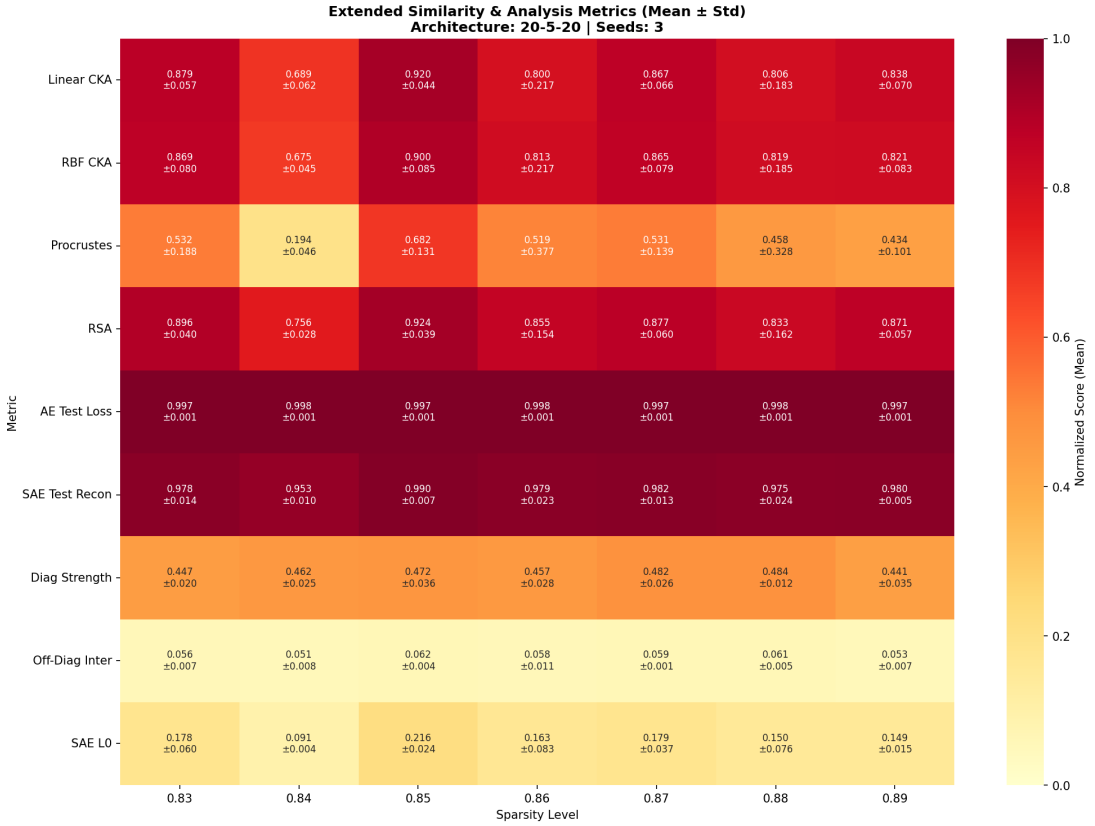


Figure 7: Extended similarity and analysis metrics across sparsities for the 20–5–20 architecture.

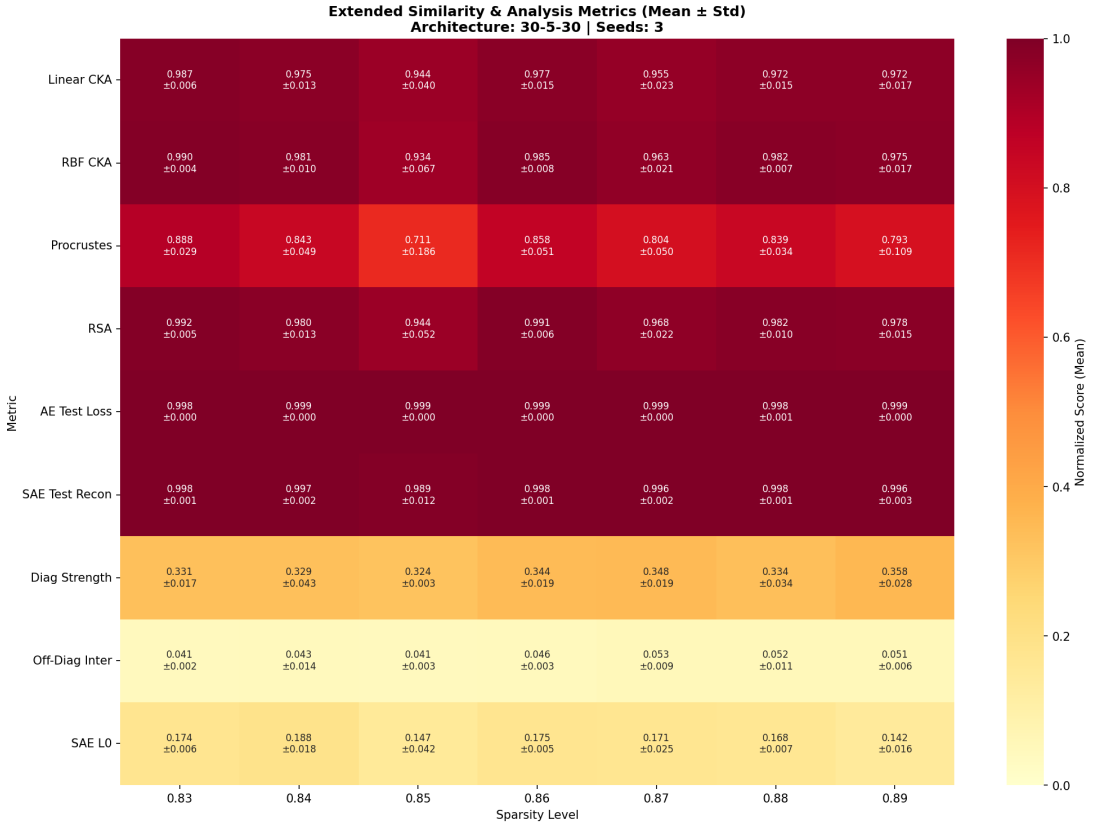


Figure 8: Extended similarity and analysis metrics across sparsities for the 30–5–30 architecture.

7 Discussion

In this paper we set out to build on the exploration of superposition in neural networks by Elhage et al. [5], testing how well their findings support the approach taken in Towards Monosemanticity [4]. Toy models showed how superposition could be artifically induced in an autoencoder by training it on sparse synthetic data, discovering a critical range between

$$(0.7 < S < 0.9)$$

. Towards Monosemanticity attempts to reverse engineer this process on language models, by training sparse autoencoders to unpack these superposed features. The large hidden dimension size with an L_1 sparsity penalty, encouraging 1-2 neurons to fire on each input was designed to make these features monosemantic i.e each one would correspond to a single interpretable concept rather than a superposed feature.

7.1 Phase Change

By training autoencoders in the critical sparsity region we were able to induce superposition, verifying it had occurred with Weight matrix plots, following [5]. Then by visualising these feature spaces in two and three dimensions we were able to shed light on how the geometry changes across this phase transition. Towards the middle of the region, interference between features increased, with the model initially encoding them orthogonally, then combining neighbouring features into antipodal pairs, and then distributing interference evenly across dimensions. This was evidenced by the blue squares in the superposition matrices growing in number and departing from the main diagonal. We measured these changes using diagonal strength and off diagonal interference. While this was indicative of the resulting shape no pattern, consistent across sparsities, emerged. The antipodal pairs that appear initially are likely the most sparsely occurring features in our data, which explains why the model could encode them in the same dimension without performance degrading. As sparsity increases the model is able to do this with more features in a variety of arrangements. However our visualisations do not indicate an obvious pattern between these arrangements and they seem to fluctuate stochastically across neighbouring sparsities. The qualitative shift in geometries is indicative of phase change occurring, confirming our second hypothesis. Deeper analysis is needed to fully understand this transition. More runs at finer sparsity granularities could reveal cyclical patterns in this encoding reminiscent of Conway's game of life [7].

7.2 Feature learning

Next we attempted to decompose these features with SAEs of mirrored dimensionality. The autoencoder is explicitly trained to reconstruct the original inputs in its final layer, as a result of its loss function being the identity function. Therefore it is encouraged to recover the original features. SAEs also do this

in their final layer but as with autoencoders their hidden representations can be learned in any representation that allows reconstruction. This experiment tested whether the representations learned in the SAE's hidden layer align with the original features. Alignment metrics between the final layer of the autoencoder and hidden layer of the SAE showed poorer performance across the critical sparsity region, suggesting that the SAE often learned a different representation than that of the Autoencoder, while still being able to reconstruct the inputs successfully. Our metrics were internally consistent but fluctuated irregularly across sparsities. This is a clear indication that the representations learned are influenced by the stochastic learning process, and so cannot be assumed to recover the same feature basis as the autoencoder. This means there is no guarantee for an SAE to learn the original features encoded however as hidden layer size increased, the representations learned performed better on alignment metrics. This suggest increasing the hidden layer dimension makes SAE's more likely to approximate the original features. This supports our third hypothesis, that a phase change in the encoding strategy is more likely to occur as sparsity crosses this critical region, and that SAEs do not reliably recover the original representations during this transition. This connects directly to critiques of the SAE approach in Towards Monosemanticity [4], where feature splitting, instability across runs, and dependence on hidden dimensionality raise concerns about whether the features extracted genuinely reflect the network's internal representations or are artefacts of the method.

7.3 Visualisations

We have demonstrated the nature of superposition is related to the stochastic process that produces it. This is evidenced by the irregularity in interference in our weight matrix, shape of feature space, and performance of alignment metrics. However further research is needed to clarify factors that lead to its emergence, beyond dimensionality and sparsity. We have contributed visualisations and metrics that facilitate interpretation of this phenomenon. Diagonal strength and off-diagonal interference generally increase across the critical region. From our visualisations we saw how these represent interference being shared more evenly across features. This is supported by the types of polytopes that emerge. 8 features into 3 dimensions produces a square antiprism, which has also emerged as a solution to the Thomson problem, that seeks a configuration of electrostatic charges with minimal interference [22]. However, these metrics assess the weight matrix, and we have seen evidence of different matrices resulting in the same representation, so they are probably not the most valuable direction of future research. Since our primary concern is with the representations actually learned by the network, future work should place greater emphasis on representations rather than weights.

7.4 Takeaways from Procrustes Analysis

We have also shown how alignment metrics can be a useful tool for assessing how representations differ, with Procrustes analysis standing out in particular. Among the metrics tested, Procrustes was the most sensitive to changes in the critical sparsity region, consistently highlighting divergence in encoding strategies even when other measures scored highly. This implies that Procrustes analysis offers a robust means of assessing when two learned representations are functionally different despite reconstructing inputs equally well.

This finding is consistent with the origins of Procrustes analysis in geometric morphometrics, where it is widely used to compare shapes by aligning them with optimal rotation, translation, and scaling [20]. Just as in morphometrics it can reveal subtle differences in biological forms, here it proves valuable in detecting subtle differences in neural representations.

Metric	**What it Measures**	**Invariances / Properties**
RSA (Representational Similarity Analysis) [11]	Correlation between pairwise dissimilarity patterns of representations.	Captures relational structure; invariant to scaling and rotation of individual features, but sensitive to noise in distance estimates.
CKA (Centered Kernel Alignment) [10]	Kernel-based similarity between representation spaces using HSIC.	Invariant to isotropic scaling and orthogonal transformations; robust across architectures and dimensionalities.
Procrustes Analysis [20]	Geometric similarity after optimal alignment of two point sets.	Sensitive to geometric transformations; explicitly aligns shapes under rotation, translation, and scaling, highlighting residual differences.

Table 1: Comparison of alignment metrics used in this study. RSA and CKA are widely used in neuroscience and machine learning, while Procrustes provides complementary sensitivity to geometric differences.

Other approaches, such as Representational Similarity Analysis (RSA) [11] and Centered Kernel Alignment (CKA) [10], are prevalent in neuroscience and machine learning respectively, and provide complementary invariances (RSA to relational structure, CKA to scaling and isotropic rotations). Our results suggest that Procrustes is especially attentive to geometric transformations, therefore there is value in including it alongside these metrics in future study of representations.

8 Conclusion

This study reproduced the phase transition described in [5], showing clear evidence of superposition emerging within the critical sparsity region. Visuali-

sations revealed a qualitative shift in geometry, from orthogonal encoding to antipodal pairs and more evenly shared interference.

When testing sparse autoencoders as in [4], we found that while reconstructions remained strong, representational alignment with autoencoders fluctuated unpredictably. Procrustes analysis was most sensitive to these divergences, suggesting that SAEs often learn alternative representations rather than faithfully decomposing superposed features.

Our results confirm the occurrence of a phase change but raise doubts about sparse autoencoders as reliable feature dictionaries. Future work should explore finer-grained experiments and improved alignment metrics to clarify the nature of superposition.

9 Future work

9.1 Parameter representation relationship

This paper has analysed the weight matrices and representations learned as superposition occurs but has failed to draw a link between the two. Bridging this gap would allow for better study and potentially the capability to steer the representations learned.

9.2 Cyclical pattern

The representations that emerge are produced stochastically but with the same random seed will occur predictably. More runs at finer sparsity granularities and zoomed in on narrow ranges could contribute valuably by illuminating the way weight matrices are tuned (e.g., order of features learned), or how these paramaterizations lead to the representations that emerge.

9.3 Importances

Hyperparameters other than sparsity have been demonstrated to affect the nature of superposition. Future work should perform similar tests with alternative functions defining feature importance (e.g., uniform, linear, step, exponential with other bases).

9.4 Alternative alignment metrics

This study compared representations using RSA, CKA, and Procrustes alignment. While Procrustes proved most sensitive to phase changes, each metric has limitations. Future work could explore hybrid or novel metrics that combine relational, kernel-based, and geometric perspectives, enabling a more complete characterisation of representational change.

9.5 Scaling to larger models

Our experiments were performed on small autoencoders with synthetic data. Extending these analyses to larger architectures or pretrained models would test whether the same phase transitions and alignment behaviours occur in more realistic settings, and whether SAEs remain viable as interpretability tools.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[2] Trenton Bricken and Neel Nanda. *Interpretability Dreams*. LessWrong. 2023. URL: https://www.lesswrong.com/posts/J2Gd7h2E6qv9Qh7nb/interpretability-dreams.

[3] Trenton Bricken et al. "Towards Monosemanticity: Dictionary Learning for Sparse Feature Discovery in Language Models". In: *NeurIPS 2023 Workshop on Mechanistic Interpretability*. 2023.

[4] Nelson Elhage et al. "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning". In: *arXiv preprint arXiv:2301.04709* (2023). URL: https://arxiv.org/abs/2301.04709.

[5] Nelson Elhage et al. "Toy Models of Superposition". In: *arXiv preprint arXiv:2209.10652* (2022). URL: https://arxiv.org/abs/2209.10652.

[6] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the manifold hypothesis". In: *Journal of the American Mathematical Society* 29.4 (2016), pp. 983–1049. DOI: 10.1090/jams/852.

[7] Martin Gardner. "Mathematical Games: The fantastic combinations of John Conway's new solitaire game 'Life'". In: *Scientific American* 223.4 (1970), pp. 120–123.

[8] Atticus Geiger et al. "Causal abstraction: A theoretical framework for model interpretability". In: *Journal of Artificial Intelligence Research* 77 (2023), pp. 517–572. DOI: 10.1613/jair.1.14156.

[9] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)* (2015). URL: https://arxiv.org/abs/1412.6980.

[10] Simon Kornblith et al. "Similarity of neural network representations revisited". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3519–3529. URL: https://arxiv.org/abs/1905.00414.

[11] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. "Representational similarity analysis—connecting the branches of systems neuroscience". In: *Frontiers in Systems Neuroscience* 2 (2008), p. 4. URL: https://doi.org/10.3389/neuro.06.004.2008.

[12] Ramón López, Jesús de la Torre, Francisco González, et al. "Fundamentals of Artificial Neural Networks and Deep Learning". In: *Applied Sciences* 14.15 (2022), p. 6578. DOI: 10.3390/app14156578.

[13] Alexander Lyzhov and contributors. *Feature Geometry in SAEs: Rotation and Reparameterisation Issues*. Alignment Forum. 2023. URL: https://www.alignmentforum.org/posts/XXXXXXXX.

[14] Jesse Mu and Jacob Andreas. "Compositional Explanations of Neurons". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020, pp. 17153–17163. URL: https://arxiv.org/abs/2006.14032.

[15] Neel Nanda. *Critiques of Sparse Autoencoder Interpretability*. LessWrong. 2023. URL: https://www.lesswrong.com/posts/HYghkQ7NQf8oTciAo/notes-on-anthropic-s-interpretability-research.

[16] Andrew Ng. *Sparse Autoencoder Tutorial*. Stanford CS294A Lecture Notes. 2011. URL: https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

[17] Chris Olah et al. "Zoom In: An Introduction to Circuits". In: *Distill* (2020). URL: https://distill.pub/2020/circuits/zoom-in/.

[18] Chris Olah et al. "Zoom In: An Introduction to Circuits". In: *Distill* 5.3 (2020). DOI: 10.23915/distill.00024. URL: https://distill.pub/2020/circuits/.

[19] Tilman Räuker et al. "Toward Transparent AI: A Survey on Interpretable Machine Learning". In: *arXiv preprint arXiv:2207.13243* (2022). URL: https://arxiv.org/abs/2207.13243.

[20] F. James Rohlf and Dennis Slice. "Rotational fit (Procrustes) methods". In: *American Journal of Physical Anthropology* 82.3 (1990), pp. 283–296.

[21] Peter H. Schönemann. "A generalized solution of the orthogonal Procrustes problem". In: *Psychometrika* 31.1 (1966), pp. 1–10. DOI: 10.1007/BF02289451.

[22] J. J. Thomson. "XXIV. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure". In: *Philosophical Magazine Series 6* 7.39 (1904), pp. 237–265.

[23] Guangyu R Yang. "Artificial neural networks for neuroscientists: A primer". In: *Neuron* 107.6 (2020), pp. 1048–1070. DOI: 10.1016/j.neuron.2020.06.009.

10 Appendix

A Methodological Details

This appendix provides full technical specifications of the models, data generation, and analysis functions described in the main Methodology section. All notation follows that introduced earlier.

A.1 Architectures

We train autoencoders of varying dimension, expressed as $n-m-n$, where n is the input and reconstruction dimensionality and m the hidden dimension. Architectures studied include:

- Main: $20-5-20$ (as in [5]), $10-5-10$, $30-5-30$.
- Small: $3-2-3$, $5-2-5$, $4-3-4$, $8-3-8$ (for 2D/3D visualisation).

A.2 Activation Function

We use the Rectified Linear Unit (ReLU):

$$\mathrm{ReLU}(x) = \max(0, x).$$

A.3 Loss Functions

For autoencoders we minimise weighted reconstruction error:

$$\mathcal{L}(x, \hat{x}) = \sum_{i=1}^{n} I_i \left(x_i - \hat{x}_i\right)^2,$$

with importance weights

$$I_i = \alpha^{\,i-1}, \quad \alpha = 0.7.$$

For sparse autoencoders, the loss is

$$\mathcal{L}_{\mathrm{SAE}} = \frac{1}{N}\sum_{i=1}^{N} \|z_i - \hat{z}_i\|^2 + \lambda \frac{1}{N}\sum_{i=1}^{N} \|h_i\|_1.$$

A.4 Data Generation

Synthetic inputs are defined as

$$x_{ij} = B_{ij}\, U_{ij}, \quad B_{ij} \sim \mathrm{Bernoulli}(1 - S), \;\; U_{ij} \sim \mathrm{Uniform}(0, 1),$$

where S controls sparsity ($0.7 < S < 0.9$ is critical [5]).

A.5 Optimiser

All models are trained using Adam [9]:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$.

A.6 Analysis Functions

The superposition matrix is defined as

$$M = W^T W + b,$$

revealing interference patterns across features [5].

A.7 Representational Alignment Metrics

To compare AE and SAE representations we employ:

- RSA [11]: $\text{RSA}(R_1, R_2) = \text{Corr}(d_1, d_2)$. Captures relational geometry, invariant to scaling/rotation.
- Linear CKA [10]: $\text{CKA}(X, Y) = \frac{\text{HSIC}(X,Y)}{\sqrt{\text{HSIC}(X,X) \cdot \text{HSIC}(Y,Y)}}$. Invariant to isotropic scaling and orthogonal transformations.
- RBF-CKA [10]: kernel $k(x, y) = \exp(-\|x-y\|^2/2\sigma^2)$. Captures nonlinear structure.
- Procrustes Alignment [21]: $\text{Proc}(X, Y) = \frac{\text{tr}(R^{*T} X^T Y)}{\|X\|_F \, \|Y\|_F}$. Tests if spaces can be aligned by an orthogonal transform.

Together, these metrics provide complementary perspectives on similarity: RSA (relational), CKA (linear invariance), RBF-CKA (nonlinear), Procrustes (rotational).

A Mathematical Functions and Implementation Details

This appendix provides additional mathematical details and functions that were implemented in our codebase but not included in the main methodology section. These ensure full transparency and reproducibility of our experiments.

A.1 Derivatives and Gradients

Gradients were computed manually for all parameters to avoid reliance on automatic differentiation.

The derivative of the ReLU activation is:

$$\frac{\partial \text{ReLU}}{\partial x} = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x \leq 0. \end{cases}$$

The gradient of the reconstruction loss with respect to the output is:

$$\frac{\partial \mathcal{L}}{\partial \hat{x}} = 2I \odot (\hat{x} - x),$$

with I the importance weighting vector.

Gradients were then propagated to compute updates for weights, biases, and hidden activations:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial z} \otimes x + z \otimes \frac{\partial \mathcal{L}}{\partial \hat{x}_{\text{linear}}}.$$

A.2 Weight Initialization

Encoder weights were initialised from a Gaussian distribution:

$$W \sim \mathcal{N}(0, 0.01^2), \quad W \in \mathbb{R}^{k \times n},$$

with biases set to zero, $b = \vec{0}$. This ensures initial activations remain small and stable.

A.3 Sparsity Metrics

To complement L_0 and L_1 norms, we compute the Gini coefficient to measure inequality in hidden activations:

$$G = \frac{2 \sum_{i=1}^{n} i \cdot a_i}{n \sum_{i=1}^{n} a_i} - \frac{n+1}{n},$$

where a_i are the sorted absolute activation values.

The L_0 and L_1 sparsity metrics are defined as:

$$\|A\|_0 = \frac{1}{N \cdot D} \sum_{i,j} \mathbb{1}[A_{ij} \neq 0],$$

$$\|A\|_1 = \frac{1}{N \cdot D} \sum_{i,j} |A_{ij}|.$$

A.4 Reconstruction Error

Reconstruction fidelity was assessed with mean squared error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\|z_i - \hat{z}_i\|^2,$$

for both autoencoders and sparse autoencoders.

A.5 Additional Analysis Functions

Beyond the superposition matrix $M = W^T W$, we computed additional diagnostics:

- **Weight norms:**
$$\|w_i\|_2 = \sqrt{\sum_{j=1}^{k}(W^T W)_{ij}^2}$$
which assess the relative strength of individual features.

- **Pairwise dot products:**
$$s_{ij} = \langle (W^T W)_i, (W^T W)_j \rangle,$$
measuring overlap and interference between feature encodings.

Together, these supplementary functions provide finer-grained diagnostics on how superposition manifests in trained networks.

B Additional Experiments

B.1 $\lambda \to L_0$ Sweep

To confirm that our sparsity penalty reliably controls the number of active neurons, we performed a $\lambda \to L_0$ sweep. Figure 9 shows the relationship between the regularisation strength λ and the resulting mean L_0. This validates that λ effectively tunes sparsity but does not determine alignment behaviour (see main text, Section X).

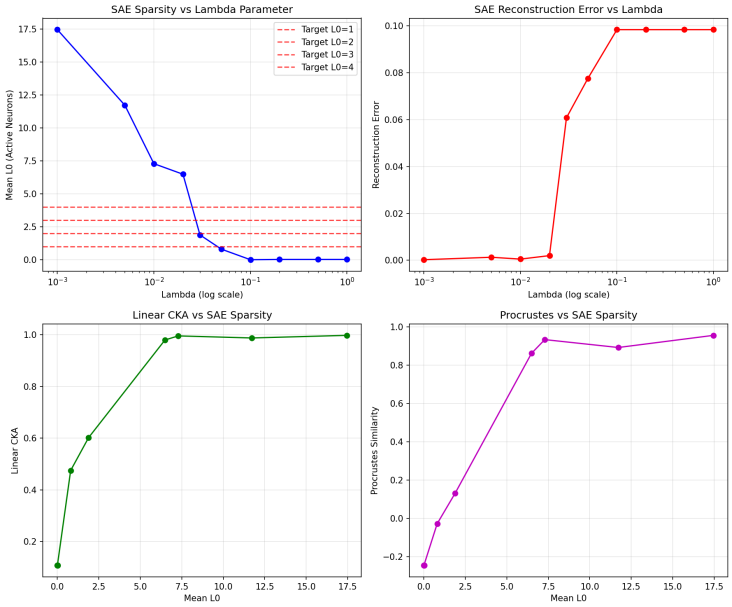


Figure 9: $\lambda \to L_0$ sweep: mean active neurons as a function of sparsity penalty.

B.2 Sweep Results

For completeness, we include the raw sweep outputs in Table 10.

Raw sweep results from `lambda_l0_sweep_results.txt`

```
Lambda -> L0 Sweep Analysis Results
==================================

Architecture: 20-5-20
Sparsity: 0.85
Target L0 values: [1, 2, 3, 4]

Detailed Results for All Lambda Values:
Lambda   Mean L0  Recon Error  Linear CKA RBF CKA  Procrustes RSA
------------------------------------------------------------------------
0.001    17.45    0.000231     0.998      0.998    0.956      0.997
0.005    11.72    0.001277     0.988      0.991    0.892      0.986
0.01     7.29     0.000489     0.996      0.997    0.934      0.998
0.02     6.49     0.001973     0.980      0.987    0.862      0.987
0.03     1.89     0.060813     0.602      0.612    0.131      0.725
0.05     0.81     0.077599     0.475      0.485    -0.028     0.538
0.1      0.01     0.098429     0.107      0.094    -0.245     0.262
0.2      0.03     0.098429     0.107      0.094    -0.245     0.262
0.5      0.03     0.098429     0.107      0.094    -0.245     0.262
1.0      0.03     0.098429     0.107      0.094    -0.245     0.262


Best Lambda for Each Target L0:
Target L0  Best Lambda  Achieved L0  Difference
---------------------------------------------
1          0.05         0.81         0.19
2          0.03         1.89         0.11
3          0.03         1.89         1.11
4          0.03         1.89         2.11
```

Figure 10: Raw $\lambda \to L_0$ sweep results.