Sparse Autoencoders for Chess Position Embeddings: An Interpretability Study of ChessLM

Ben Hull https://bluehood.github.io/

Abstract

Mechanistic interpretability techniques developed for large language models have shown that sparse autoencoders (SAEs) can decompose polysemantic neurons into human interpretable features. We investigate whether this approach transfers to ChessLM, a transformer-based model trained to produce embeddings of chess positions. We train a single 0.5M-parameter Top-K SAE on activations from the middle layer of ChessLM and evaluate the resulting 1024 features along three axes: interpretability, behavioural relevance, and geometric organisation. Manual inspection identifies clearly interpretable features corresponding to concrete chess concepts, such as Black Knights, endgame open space, and active White pieces, some of which appear sensitive to game phase without explicit supervision. Scaling this analysis with an LLM judge across all features shows that such clean interpretability is the exception: roughly 79% of features receive the lowest interpretability score, with only around 10% scoring as clearly interpretable. Feature clamping experiments, in which we amplify or suppress individual features and observe the effect on similarity search, provide mixed evidence that these interpretations are causally load-bearing rather than purely descriptive. Finally, we find that features with related interpretations do not reliably cluster in the SAE's decoder space, suggesting the dictionary has not learned a clean semantic taxonomy at this scale. Together, these results suggest SAEs can extract genuinely interpretable and behaviourally relevant features from a domain-specific chess model, but that a modestly sized, single-layer dictionary captures only a fraction of the model's underlying concepts cleanly, with the remainder likely entangled in superposition.

Keywords: Mechanistic Interpretability, Monosemanticity, Large Language Models, Chess

1 Introduction

Mechanistic interpretability aims to gain an understanding of complex neural networks by segmenting them into components that more readily understood than the whole [1]. Anthropic showed this was possible for a one-layer transformer by decomposing the language model using a sparse autoencoder. This resulted in a large number of features that had human readable interpretations.

By understanding the purpose and function of each feature/component, the theory is that researchers will be able to understand the behaviour of the entire network. Unfortunately, the building blocks of a neural network – the neurons themselves – are not human interpretable; they consist of a set of weights and biases which are just a collection of numbers. Indeed neurons are polysemantic meaning they capture the meaning of a number of unrelated concepts [2]. As we shall see later, models attends to many different features in complicated ways. For example, in the domain of Chess, a single neuron may contribute to

the calculation of piece mobility, tactical motifs and pawn structure in a way which is not directly human interpretable.

We apply the technique of extracting interpretable features from Claude 3 Sonnet using Sparse Autoencoders (SAE) [3] to ChessLM, a Transformer-based model designed to learn rich, contextual vector representations (embeddings) for Chess positions [4]. We will find that the technique is useful in extracting features with a human interpretable function across the network. Indeed we can interpret a number of these features as subproblems on a chess board; for example, focusing on the White pieces pawn structure or the location of the White king.

1.1 Dictionary Learning

Anthropic's approach to understanding Cladue 3 Sonnet is based on two hypothesises: the linear representation hypothesis [5] and the superposition hypothesis [6]:

- **Linear Representation Hypothesis**: Neural networks represent meaningful concepts (features) as directions in their activation space.
- **Superposition Hypothesis**: Neural networks use almost-orthogonal directions in high dimensional spaces to represent more features than their are dimensions.

If we accept these hypotheses, the standard approach is to use **Dictionary Learning**. In particular, an approximation of dictionary learning called Spare Autoencoders (SAE) is effective [1].

1.1.1 Sparse Autoencoders

Sparse autoencoders (SAE) is a variation of traditional autoencoders designed to learn compressed representations of data by keeping most of their internal neurons turned off. The SAE consists of three main components:

- **The Encoder**: Takes the input data and compresses it into a hidden layer (latent space).
- **Hidden Layer**: A layer that is larger than the dimensionality of the input data (overcomplete) however, it is restricted on how many neurons can fire at once.
- **The Decoder**: Takes the sparse representation from the hidden layer and aims to reconstruct the original input as accurately as possible. In our implementation, we take the top K most significant activating neurons from the sparse representation to make the reconstruction task produce a sparse vector space.

By forcing sparsity, that is, the top K (32 in this work) of neurons are dropped at a given time, forces the network to learn from the most significant features. It can only reconstruct the representations from the most meaningful hidden state parameters. The model is trained to minimise a combination of (a) the reconstruction error and (b) a L1 regularisation penalty on the feature activations. The L1 penalty causes the model to prioritise the most significant features, which incentivised sparsity.

1.2 SAE Setup

In this work, we focused on applying SAEs to activations halfway through the model (in the middle layer). ChessLM is a six layer deep Encoder Transformer and therefore we read the activations at the end of the fourth layer. We chose this level of the model because we believed that the model was likely to contain abstract features, some of which would be directly interpretable.

We trained a single SAE with $525,568$ (0.5M) parameters on approximately 10M Chess positions. The number of training epochs was discovered by training the model for a significant period of time, and selecting the most appropriate number of epochs which minimised the validation loss without the model exhibiting signs of overfitting.

2 Feature Interpretability

In the previous section, we described the approach we took to training a sparse autoencoder (SAE) on ChessLM. We now want to understand whether the SAE has produced interpretable features which help us understand the model behaviour. We'll analyse the results to see whether the SAE features are interpretable and whether they can help us understand the model behaviour. We'll start with the more straightforward features that we looked at and show that they track key concepts in Chess positions, before moving onto more complex features. We'll then scale the analysis of different features by providing positions, with feature overlays, to an LLM to classify them according to a custom rubric.

2.1 Examples of Interpretable Features

Here we'll look at a few features and argue that they are genuinely interpretable. We want to understand whether the features exist, and understand whether the majority of features are interpretable or not. We will give evidence that our interpretations are good descriptions of what features in the network represent.

- Black Knights: the feature focuses on Black Knights in the position.
- Open Space in the Endgame: focused on open spaces not occupied by pieces in endgame positions.
- White Position: Focuses on active White pieces and pawn structure.

In this paper, we show representative examples from the top twelve positions in our dataset, ranked by how strongly they activate features. We constructed the following rubric for scoring how a feature's description relates to the position on which it fires. We then manually assessed positions on that rubric.

- 0 – The feature is completely irrelevant and does not show any meaningful relationship
- 1 – The feature shows clear meaning in the position but the interpretation is not clear
- 2 – The feature shows significant meaning in the position and the interpretation is clear

2.1.1 Black Knights

We can clearly see that the Black Knights are captured by this feature Figure 1 shows four different positions on a Chess board and the activations of the squares on each board, with blue showing a negative activation and red showing a positive activation. As can be seen a positive red activation shows for squares occupied by a Black Knight. Its greatest activations are all essentially references to the positions of the Black Knights. This is present across many different stages of the game, including endgame and middle-game positions, suggesting this feature is game-phase invariant.

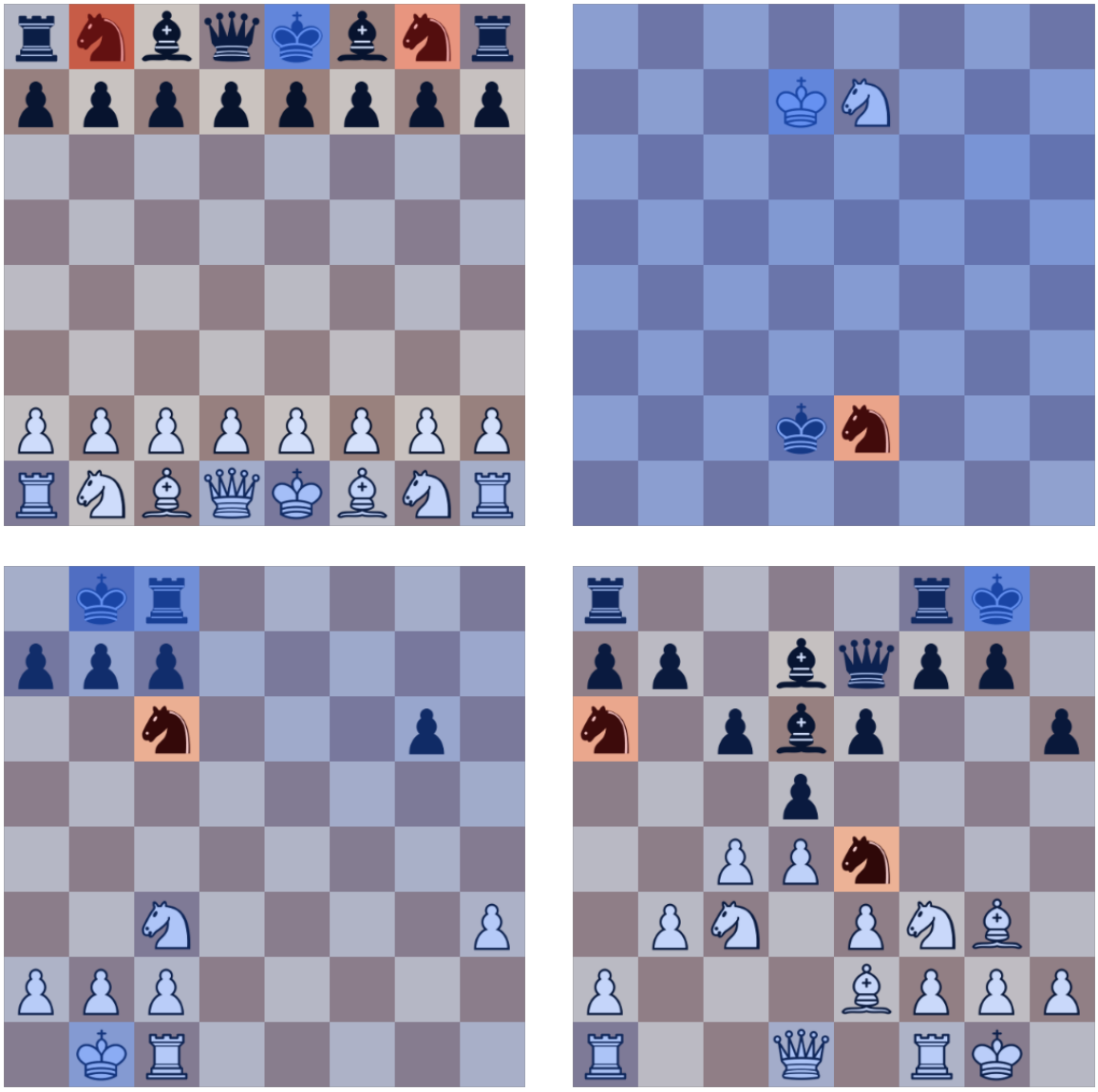


Figure 1: Feature 426 applied to four positions showing that the feature focuses on Black Knights. Black Knights have positive activations across different positions, shown by the positive red activation (as opposed to the negative blue activation shown elsewhere).

2.1.2 Open Space in the Endgame

Feature 253 clearly focuses on open squares in positions towards the end of the game (when there are fewer pieces and pawns on the board). Figure 2 shows the application of feature 253 to three endgame positions and the opening position. For the three endgame positions the model clearly focuses on open squares and actively ignores squares taken by pieces of both colours. This suggests that this feature is focused on open space in endgame positions.

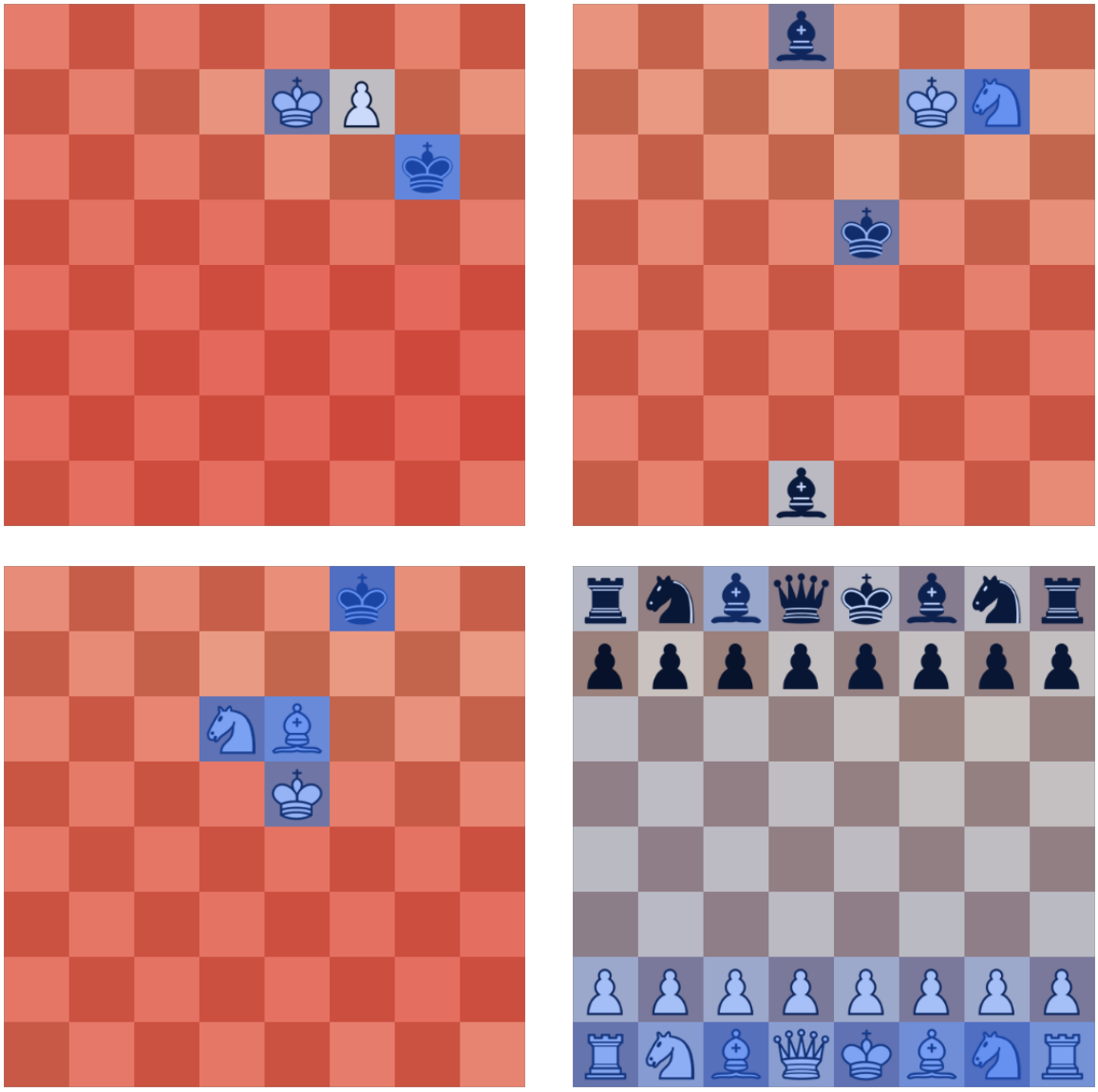


Figure 2: Feature 253 applied to four positions showing that the feature captures non-occupied squares in endgame positions. Open positions have positive activations (shown in red) whereas pieces and actively ignored (shown in blue for negative activations). The final position shows the opening position of a chess game. Interestingly, the feature does not activate in the opening position, suggesting the model has actively learnt to identify the phase of the game, and has specific features to analyse different phases of the game.

Interestingly, the same activation cannot be seen for the opening position – the feature is deactivated and does not focus on the open spaces in the position. This suggests that the model has learnt to identify different phases of the game; in this case, the endgame.

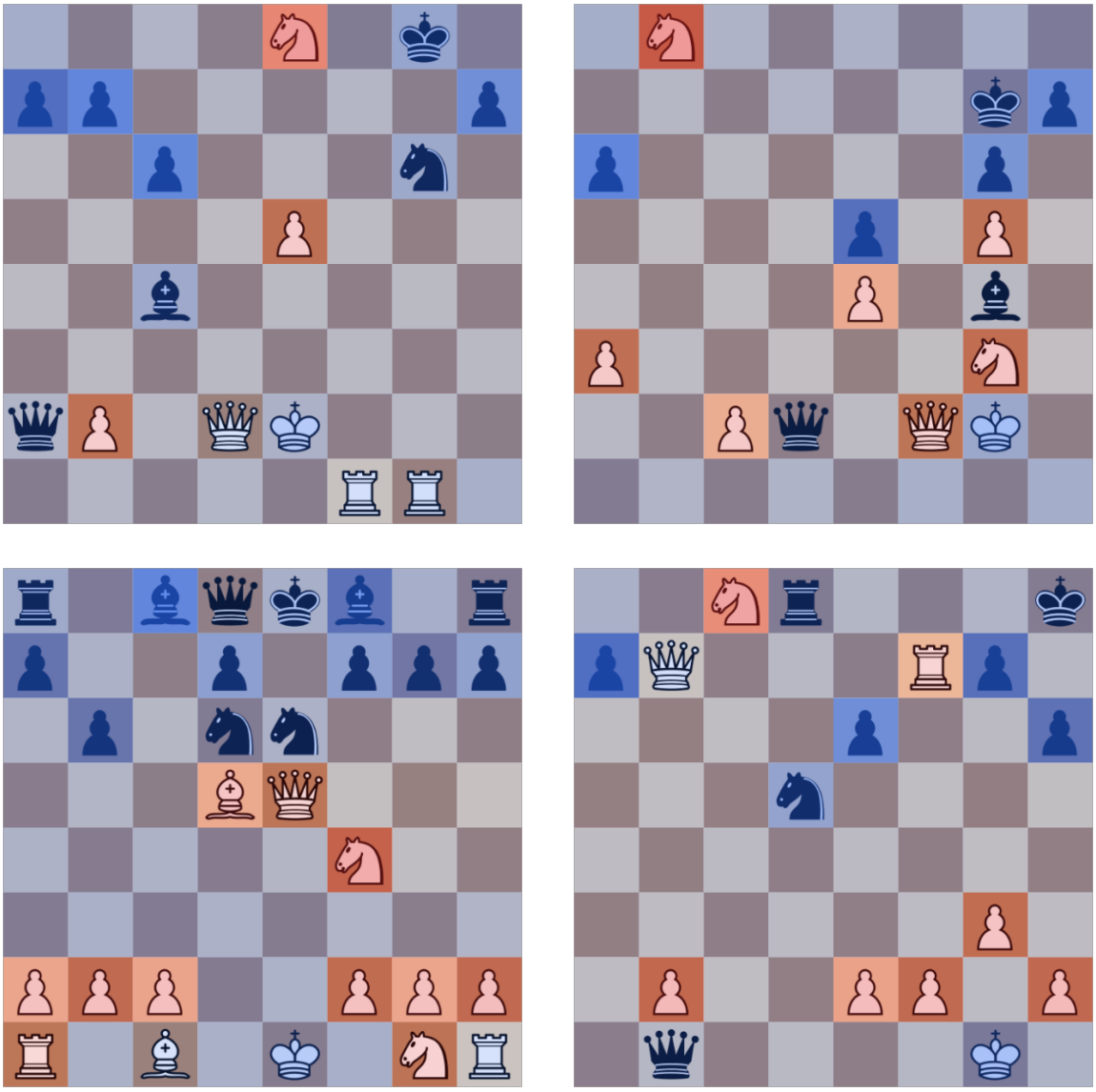


Figure 3: Feature 188 applied to four positions showing that the feature prioritises key White pieces in the position. Across different phases of the game, but particularly in the middle game, the feature identified key White pieces in the position, and White pawns. The feature actively ignores Black's position and White King.

Additionally, this suggests that different features are activated for different stages of the game.

2.1.3 White Position

We analyse a more general features here; feature 188 picks up on key pieces in White position, including pawns. The feature tends to pick up on White's key pieces that are

active and actively ignores Black's position. Figure 3 shows the application of feature 188 to several positions. In each case we can see that key White pieces show a positive activation across different stages of the game, but the effect is particularly prevalent in the middle game, with lots of active pieces.

2.2 Scaling Interpretability

With 1024 features it becomes difficult for manual review to cover all features. Instead we use an LLM, in this case Gemini (gemini-3.1-flash-lite), to analyse the features instead. We provide the top four positions by activation of the feature, with the red-blue overlay to give the model context of which squares are activated by the feature. This made it much easier to scale the analysis of all the features in a consistent fashion.

2.2.1 LLM-as-Judge Methodology

The LLM was tasked to score each feature, subject to the positions it was given, according to the rubric outlined in Section 2.1. The task was to process each feature and score it either zero, one or two; zero being a score indicating the feature was not interpretable at all and two being the feature had a clear and consistent interpretation.

This "LLM-as-judge" approach is attractive because it lets us apply a single, consistent rubric across all 1024 features in a fraction of the time manual review would take – a single researcher reviewing four positions per feature across 1024 features would represent a substantial undertaking, whereas the LLM can process the full set of features in a single pass. The trade-off is that we substitute a fast, scalable judge for a slower, arguably more reliable one; we discuss the implications of this trade-off below.

Table 1 shows the results of this procedure.

Metric	**Value**
Total features	1024
Score 0	812
Score 1	111
Score 2	101

Table 1: Feature Score Distribution

Indeed it appears that the majority of features (812 or 79.3%) are uninterpretable according to the LLM. Our SAE has failed to produce genuinely interpretable features roughly four out of five times. The remaining features scored one roughly 10.8% of the time and scored two 9.9% of the time, suggesting a modest fraction of our features are interpretable.

2.2.2 Score One from Score Two Features

It is worth unpacking what separates a "clear meaning, unclear interpretation" feature (score 1) from a "clear meaning, clear interpretation" feature (score 2), since the difference is not simply a matter of degree. Score-2 features, such as feature 426 (Black Knights) or feature 253 (Open Space in the Endgame), tend to share two properties: they fire on a well-defined, low-level visual property of the board (a particular piece, a particular colour,

or occupied versus empty squares), and they do so consistently across the four sampled positions regardless of game phase or material balance.

Score-1 features, by contrast, often show a directionally consistent pattern; for example, consistently activating on one side of the board or on one colour's pieces, but without a single unifying explanation that covers all four sampled positions. In several cases we observed during manual spot-checking, a score-1 feature would activate cleanly on three of four positions but pick out a seemingly unrelated square or piece in the fourth, which is enough to prevent a confident single-sentence description while still reflecting an underlying pattern. This suggests that many score-1 features are not simply noise, but partially-formed or entangled versions of a more interpretable concept. Consistent with the superposition hypothesis, in which a single direction in activation space is shared between more than one true underlying feature.

2.2.3 Limitations of Automated Scoring

Several caveats apply to the distribution in Table 1, and to LLM-based feature scoring more generally.

First, the LLM sees only four positions per feature, sampled by activation strength. This biases the evaluation toward a feature's most extreme behaviour and may miss more subtle or context-dependent activations that a broader sample would reveal. A feature that is genuinely interpretable but only fires weakly, or fires on a rarer board configuration, is more likely to be under-represented in its top-four activating positions and therefore under-scored.

Second, we used a single LLM (Gemini) with a single pass per feature, rather than an ensemble of models or repeated sampling. Without repeated trials we cannot establish inter-rater reliability, and it is possible that some features scored 0 would be assessed differently, either more or less favourably, by a different model, a different prompt, or a repeated run of the same model. LLM judges are also known to be sensitive to prompt phrasing and to exhibit their own biases (for instance, a tendency to under- or over-report uncertainty, or to anchor on superficial visual patterns such as symmetry or colour balance rather than chess-specific semantics). We did not measure agreement between the LLM's scores and a human rater on a held-out sample, which would be a natural extension of this work and would let us calibrate how much to trust the reported 79.3% uninterpretable rate.

Third, the rubric itself asks for a single scalar judgment of interpretability, which collapses two distinct failure modes into a single "0" category: features that are genuinely uninterpretable (noise, or so entangled with unrelated concepts that no consistent story emerges), and features that may be interpretable but whose interpretation requires chess-specific reasoning beyond simple piece or square identification. For example, features tracking a tactical motif, king safety, or piece coordination, all of which appeared in our manual feature notes (e.g. Feature 140, focused on king safety, or Feature 102, focused on space and attacking pieces). An LLM without strong chess understanding, or without being explicitly prompted to reason about tactical and positional concepts rather than only piece identity, may systematically under-score this second category, since verifying a claim like "this feature tracks king safety" requires evaluating the position's tactics rather than simply reading off occupied squares. This suggests the reported interpretability rate may be a conservative

lower bound with respect to visually simple features (pieces, colours, occupancy) and a more substantial underestimate with respect to higher-level positional or strategic features.

2.2.4 Interpreting the Uninterpretable Rate

Taken together, the finding that roughly four in five features are not interpretable is informative about our SAE architecture and training regime rather than simply a negative result. A single 0.5M-parameter SAE trained with a fixed Top-K of 32 is a fairly coarse dictionary relative to the six-layer ChessLM model it is decomposing; the dictionary learning literature suggests that increasing dictionary size tends to reveal progressively finer-grained, more monosemantic features through a process sometimes called feature splitting, where a single coarse feature in a small dictionary resolves into several more specific features in a larger one. It is plausible that many of our score-0 features are early, entangled versions of concepts that would separate into cleanly interpretable features under a larger dictionary, a different sparsity level, or activations sourced from a different layer of the network.

This is broadly consistent with the results in Section 7, where we find that features close together in decoder space do not share similar interpretations. A dictionary that has not yet fully separated overlapping concepts would be expected to show exactly this kind of pattern: nearby directions correspond to still-superposed combinations of features rather than to cleanly split, independently interpretable ones.

We've seen three examples of genuinely interpretable features on the chess board and different ways in which the model generates embeddings; from simple one piece features to more general features, such as Open Space. We've also seen that the use of LLM's to analyse positions scales well but has several drawbacks and is likely a underestimate on how interpretable the features are. In the next section we look to understand exactly how these features influence the behaviour of the model; do the interpretations we've made translate to the behaviour of the model itself.

2.3 Behavioural Influence

Next we want to demonstrate whether the interpretation of features accurately describe their influence on model behaviour. As Anthropic did for Claude Sonnet [3] we want to clamp specific features to artificially high or low values and observe the effect it has on the ChessLM model behaviour. We find limiting evidence that the interpretation of the feature line up with how they are used by ChessLM.

We clamp a given feature to some multiple of its activation value. We then load a chess position and perform a similarity search through a corpus of 10 thousand randomly selected positions of the 175 thousand game positions. We can then compare the positions most similar to the original position, with one of the features clamped to an artificially high or low position.

We see that clamping the Black Knights feature (feature 426) to 10 times its maximum activation causes the model to register positions with similar Black Knight more highly.

The reference position in Figure 4a shows an endgame position with one Black Knight (with positive activation). Ordinary similarity search with no clamping Figure 4b shows the same knight instead placed on the edge of the board, suggesting that the feature has not been considered above other competing features for this position. Many of the other aspects of the position are similar to the search position: the position is an endgame, pawn structure is similar and material is equal. This suggests that the ChessLM model is successful in identifying similar looking positions.

Clamping feature 426's activation to 10 times the original strength appears to make the model weight the position of the Black Knight more highly. Figure 4c shows an endgame not dissimilar from the search position. In this instance, the model has weighted the position of Black's Knight more highly, with the Knight placed in a similar position to search position (albeit not identical). This suggests that clamping the feature to an artificially high value may make the model lean on the feature more when finding similar positions (or more generally generating an embedding).

We also clamp feature 426 to an artificially low value. In this case the top most similar position does not even contain a Black Knight, suggesting the model is less concerned with Black Knights when this feature strength is weakened.

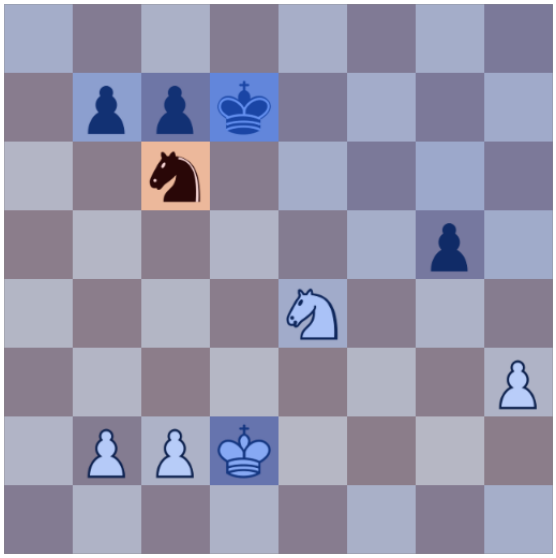
We also do the same for feature 253: Open Space in the Endgame. We find much more limiting evidence that the feature is used in the way it has been interpreted.

We use the same search position as a in Figure 4; an endgame position, with two knights and near equal pawn structure. The ordinary search with no clamping results in a similar position with similar empty space in the position, with roughly similar pawn structure to the search position. Additionally, the model has found an endgame position, suggesting the model is effective in identifying Chess positions with a similar phase in the game.

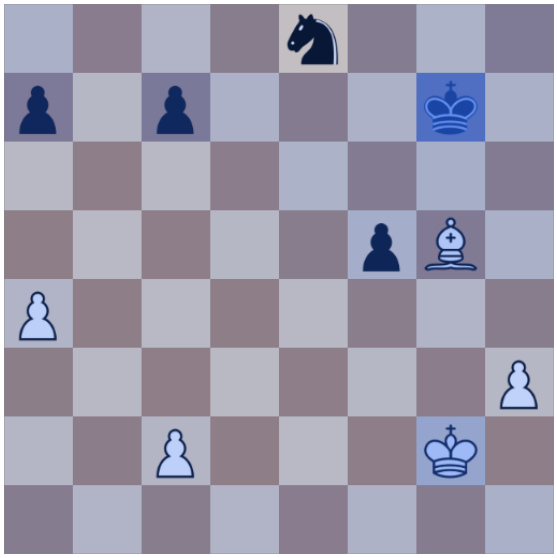
Clamping the strength of feature 253 to +10, makes the model focus heavily on open space in the endgame and also pawn structure. Figure 5c shows the result of clamping to this artificially high value and the model does appear to value the open space, as interpreted, but this is less clear when compared to Black Knights feature. Interestingly, the model has selected a top position with Queens on the board rather than two pieces, suggesting the model is prioritising empty squares rather than material.

However, with empty space being a rather abstract feature to analyse its difficult to see exactly how the model is prioritising this feature. One crude method is to calculate the number of pieces on the board; 10 for the search position and 11 for the clamped position. This suggests there is some similarity between the positions in terms of open space, but this makes no claim to the nuance of the position, such as piece placement and pawn structure. Its clear from a visual inspection, however, that the open space is similar across the two positions. Black and White both have a set of connected pawns on the left-hand side of the board and disconnected pawns on the right, with Black's King being centrally positioned and White's differing slightly in placement between the two positions. Black and White's pieces are also centrally positioned in each case. This similar piece placement means that the open space is similar in each position, however, its difficult to quantify how much of this is down to feature 253 or whether other features in the model are contributing.

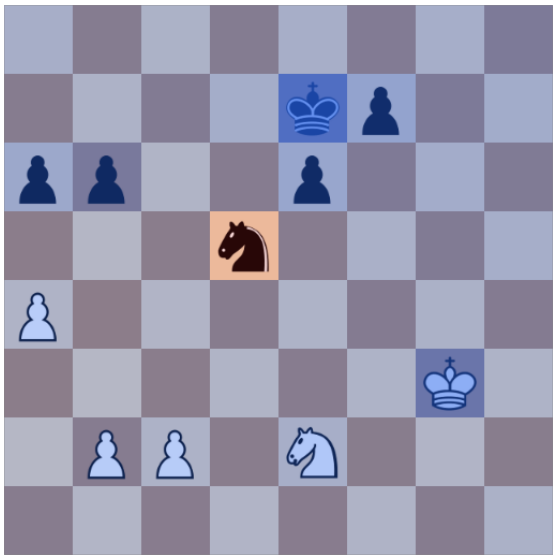
When clamping the feature to -10 we see that open space is not considered highly by feature 253 and that it has been actively suppressed. The pawn structure is similar to the search position but the number of pieces on the board differs for both sides. It appears that



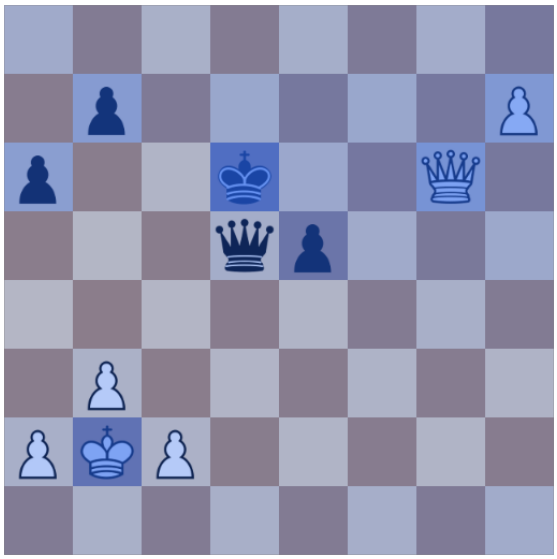
(a) Search Position



(b) Ordinary search with no clamping



(c) Search with 10 times the activation

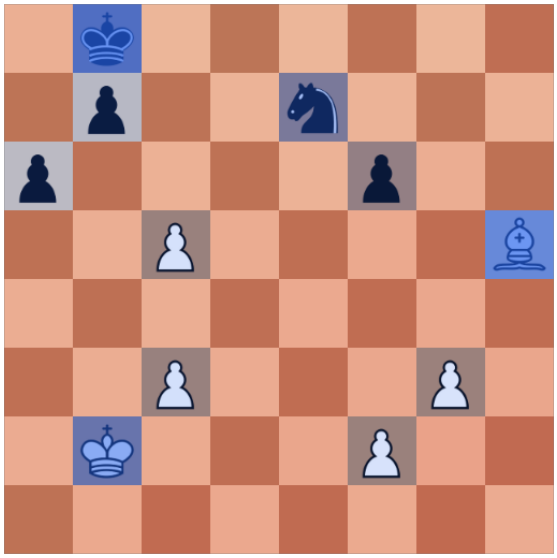


(d) Search with -10 times the activation

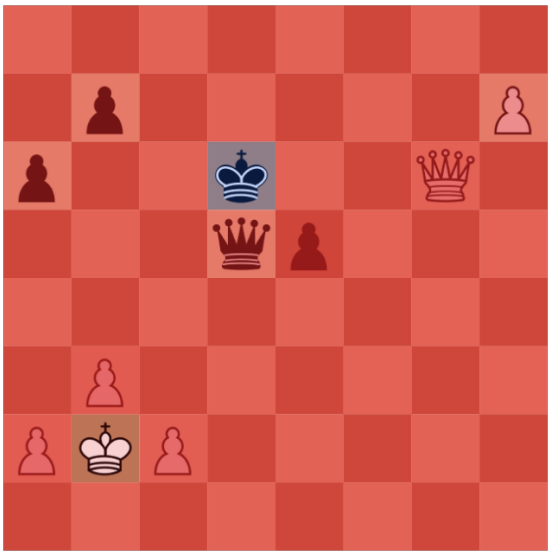
Figure 4: This shows the top position in the similarity search between 4a and our corpus of positions for feature 426. We see that in 4b we search with no clamping and the knight is not in a similar location to the search position, residing instead on the edge of the board. When increasing the activation in 4c we see that the knight is more similarly positioned to the search position, as would be expected. In 4d with a clamp of -10 we see that there is no knight in the position, suggesting that this feature has been under-represented after clamping.

feature 253 (Open space in the Endgame) is not contributing significantly to the similarity evaluation of the position; instead other features are considered above it. Interestingly, the

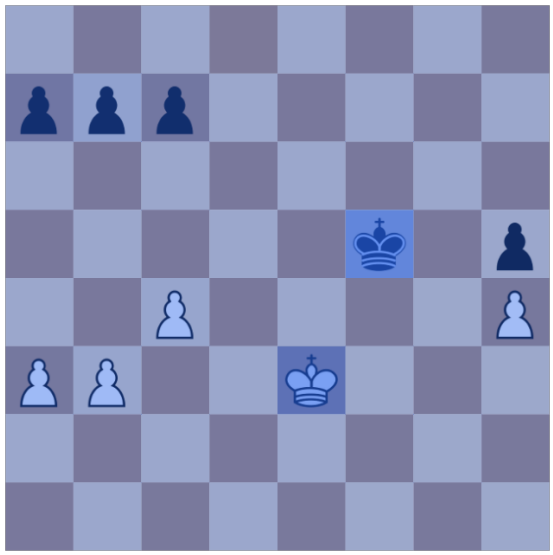
(a) Search Position



(b) Ordinary search with no clamping



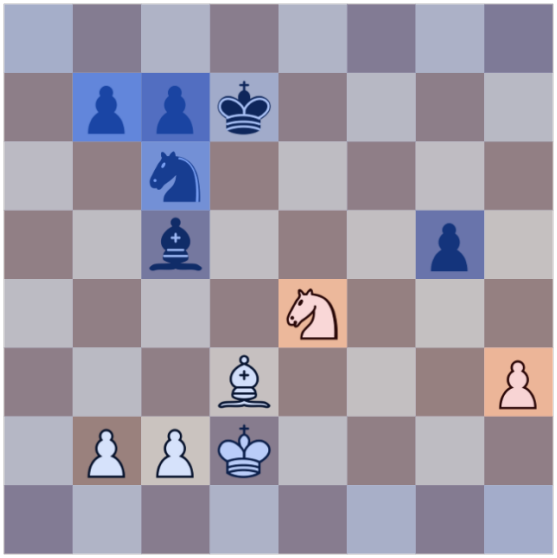
(c) Search with 10 times the activation



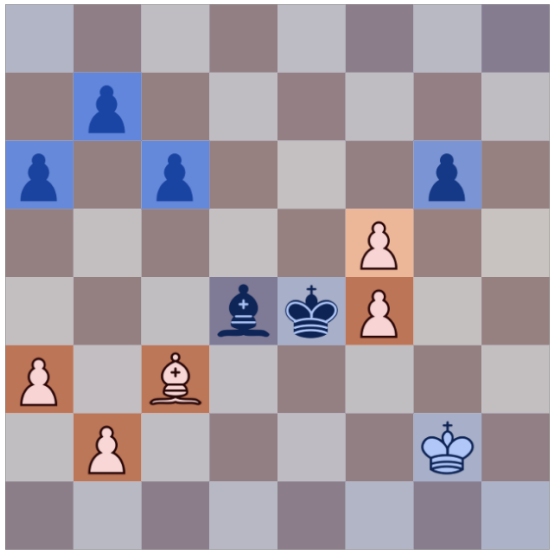
(d) Search with -10 times the activation

Figure 5: This shows the top position in the similarity search between 4a and our corpus of positions for feature 253. We see that in 4b we search with no clamping and the position shows some similarity in open space to the search position. When increasing the activation in 4c we see that the open space in the position is similar, but it is difficult to quantify that similarity. In 4d with a clamp of -10 we see that the feature has no positive activations and is not contributing to the evaluation. Despite this the model is still showing positions with similar open space and general game structure. This suggests that other features, or combinations of features, are evaluating open space.

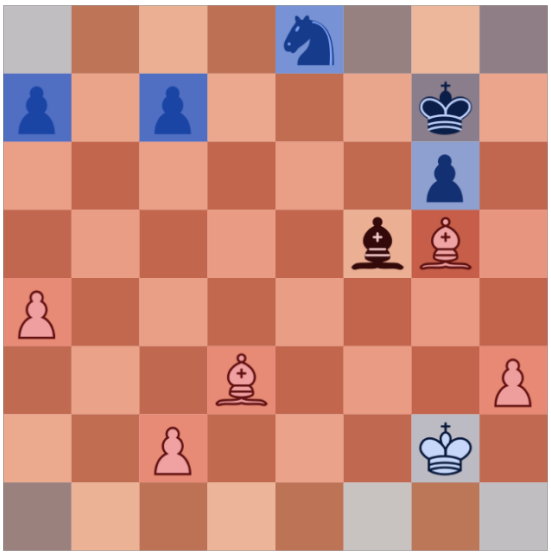
position does have open space similarity with the search positions, suggesting that other features, or combinations of features, are accounting open space in the position.



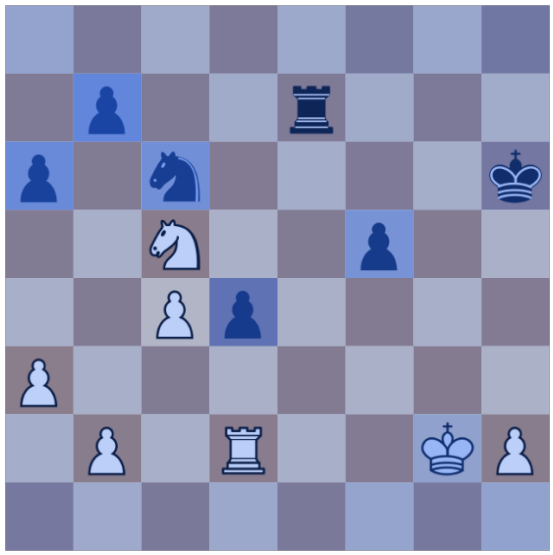
(a) Search Position



(b) Ordinary search with no clamping



(c) Search with 10 times the activation



(d) Search with −10 times the activation

Figure 6: This shows the top position in the similarity search between 6a and our corpus of positions for feature 188. We see that in 6b we search with no clamping and the position shows some similarity in the white position. When increasing the activation in 6c we see that the white position has some similarities with the original position. However, the most noticeable difference is that the model also activates empty space in the position and even Black's bishop. In 6d with a clamp of -10 we see that the feature has no positive activations and is not contributing to the evaluation. Despite this the model is still showing positions with similar White positions, particularly in pawn structure. This suggests that other features, or combinations of features, are evaluating the White position.

We now look at feature 188: White Position. This feature appears to activate on White's pieces and positions whilst actively ignoring Black's pieces. In Figure 6a we see the search position, with White's pieces activated, albeit slightly less strongly than in other positions we've seen.

With an ordinary search Figure 6b we observe a similar position, particularly in terms of the pawn structure, with both Black and White having pawn islands on each side of the board. White's pieces are also activated in the position whilst White's (and Black's) King are showing a negative activation. Additionally, Black's pieces are not activated, as seen in our previous analysis.

Clamping the feature to an artificially high value of ten times the activation strength (Figure 6c has some interesting results. As before, we see that Black's pieces have a negative activation. However, increasing the activation appears to have changed the effect of the feature. Whilst White's pieces and pawns have the strongest activation, the feature is now also activating on open squares more strongly. This suggests that the feature is also incorporating some form of open square analysis that is not observable under normal activation strengths.

Searching with minus ten times the activation strength (Figure 6d) shows a relatively similar position to the search position. Both sides are roughly equal in material with similar pawn structures to the original position. Both sides have active Knights and Rooks. Black's pieces still maintain a negative activation, however, White's pieces now no longer have positive activations as in the original position. Indeed White's pieces and pawns are no longer focused upon by the feature. This is to be expected with the feature being under represented. Despite this a roughly similar position has been returned by the search, suggesting that other features, or indeed combinations of features, are accounting for White's position.

We've seen that the interpretation of our features roughly translates to the behavioural implementation by our model. When activating these features more strongly the model tends to value the feature more highly when searching for similar positions. This is clear in the more simplistic features we have analysed, but can be more complex for other positions.

We now move on to understand local neighbourhoods of features: do features that are closer together in the decoder space have similar meanings and interpretations?

3 Feature Neighbourhoods

We survey local neighbourhoods of feature 188 (White Position) for our 0.5M SAE. We look at the SAE decoder weights and use cosine similarity to measure similarity. We find that there is little grouping of similar concepts in the decoder space; features closer in the embedding space do not appear to emphasise the same chess positions or motifs.

3.1 Methodology

For a seed feature i, we compute the cosine similarity between its SAE decoder column $\mathbf{d}_i \in \mathbb{R}^{256}$ and every other decoder column $\mathbf{d}_j$ in the dictionary, ranking features by

$$\cos(\mathbf{d}_i, \mathbf{d}_j) = \frac{\mathbf{d}_i \cdot \mathbf{d}_j}{\mathbf{d}_i \, \mathbf{d}_j}.$$

We take the top 25 nearest neighbours of feature 188 by this measure and project the resulting 256-dimensional decoder vectors into two dimensions with t-SNE for visualisation, labelling each point with our manually assigned interpretation (or "NA" where no consistent interpretation was found).

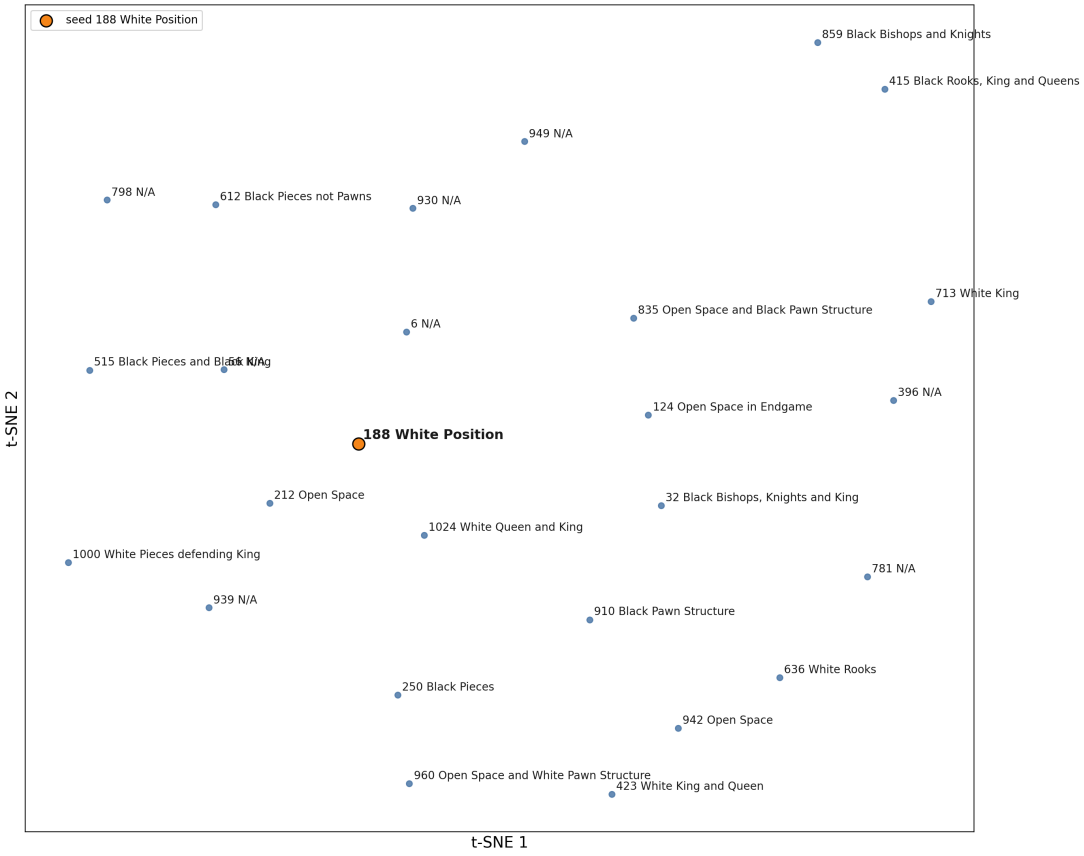
We use decoder-weight cosine similarity rather than, say, co-activation statistics because it directly asks the question: *do two features write into the residual stream in a similar direction?* This is a natural operationalisation of "similar meaning" under the linear representation hypothesis, since if two features corresponded to closely related concepts we would expect the model to represent them with similar (though not necessarily identical) output directions.

Figure 7 shows the neighbourhood of the top 25 most similar features to feature 188 (the White Position). Each of the features has been labelled with its interpretation, with "NA" for features without a clear interpretation. We observe that feature neighbourhoods do not tend to group features together with similar interpretations; indeed, sometimes close features have significantly different interpretations. For example, feature 124 - Open Space in the Endgame is ranked closely with feature 32 - Black Bishops, Knights and King, which are different aspects of a chess position.

3.2 Quantifying Neighbourhood Structure

The visual impression from Figure 7 is backed up by the underlying similarity statistics. The neighbourhood around feature 188 in our 1024-feature Top-K SAE does not exhibit a clean semantic taxonomy. Nearest-neighbour cosines for this seed are only modest (approximately 0.19–0.34), while the global distribution of pairwise decoder cosines is near zero on average (mean ≈ 0.004, 95th percentile ≈ 0.14), so "nearest neighbour" here typically means slightly less orthogonal rather than clearly synonymous.

This global near-orthogonality is itself an important sanity check: it confirms that the SAE has, on the whole, succeeded in spreading features out across the available directions rather than collapsing many features onto a small number of shared directions, which would be a sign of a badly under-trained or degenerate dictionary. The near-zero mean cosine is broadly consistent with what we would expect from a dictionary that has learned close to 1024 almost-orthogonal directions in a 256-dimensional space, as the superposition hypoth-



(a) Search Position

Figure 7: The top 25 most similar features according to the cosine similarity of decoder output weights. The 256 size embedding is reduced to a two dimensional plot using t-SNE for dimensionality reduction. Each feature is plotted alongside its interpretation with 'NA' representing features with no clear interpretation. There is little evidence that similar features in the space have similar interpretations; indeed closer features do not have similar interpretations. For example, features relating to White positions are not close in the embedding space, like we may expect if local neighbourhoods represent similar features.

esis predicts. What it does *not* confirm is that the small excess similarity between nearby features corresponds to shared semantic content, and Figure 7 suggests it largely does not.

Moreover, features grouped by our hand labels (e.g. open-space or white-royalty motifs) have only weak mean pairwise similarity in decoder space, with occasional exceptions such as features 415 and 859 ($\cos \approx 0.48$). This is expected rather than evidence that ChessLM lacks organised representations: decoder cosine measures whether features write similar directions into activation space, whereas our labels describe activation patterns ("fires when black pieces are present"), and these need not coincide. Two features can fire in a strongly correlated way – for instance, always co-occurring in king-safety-relevant positions – while still writing into near-orthogonal output directions, if the downstream computation reads

them out independently. Conversely, two features with similar decoder directions need not fire on visually or conceptually similar positions if the model uses that shared direction for two different purposes at different points in the network's computation.

3.3 An Activation-Based Alternative

Given this mismatch between decoder geometry and hand-labelled semantics, a natural extension – which we leave to future work – is to define neighbourhoods using activation statistics rather than decoder weights: for example, ranking features by the Pearson or Jaccard co-activation of their firing patterns across our corpus of positions, or by the correlation of their activation *strength* conditional on both features being active. Such a measure would more directly test whether "features that fire together" tend to share an interpretation, which is a distinct (and arguably more behaviourally relevant) question from whether they *write* in similar directions. It is plausible that activation-based neighbourhoods would show tighter semantic clustering than the decoder-weight neighbourhoods reported here, since our hand labels are themselves derived from activation patterns rather than decoder geometry.

3.4 Why Flat Neighbourhoods Are Expected at This Scale

Our SAE is also small and relatively coarse, so features are more polysemantic and neighbourhoods flatter; complementary concepts (white material vs. black material, occupancy vs. open space) can sit nearby geometrically even when their interpretations look unrelated. This is consistent with the broader finding from Section 2.1 that roughly 79.3% of features scored zero on our interpretability rubric: if a large majority of features are themselves entangled combinations of multiple underlying concepts, we should not expect their decoder directions to cluster cleanly by any single semantic axis, since each direction is doing double or triple duty. A larger dictionary that more fully resolves feature splitting – separating, for instance, "White Position" into more specific sub-features for White's king safety, White's pawn majority, and White's piece activity – would be a natural next step to test whether tighter neighbourhood structure emerges once individual features are less overloaded.

Finally, t-SNE preserves only relative similarities among the selected points, and when those similarities are uniformly weak it will scatter same-theme labels, making 2D distance a poor proxy for relatedness. We note this as a visualisation caveat rather than a claim about the underlying geometry: the quantitative cosine statistics above, not the two-dimensional projection itself, are the primary evidence for the lack of clean neighbourhood structure.

4 Conclusion

In this paper we applied sparse autoencoders to ChessLM, a transformer-based model trained to produce embeddings of chess positions, with the aim of understanding whether the mechanistic interpretability techniques developed for large language models [1; 3] transfer to a much smaller, domain-specific model. We trained a single 0.5M-parameter Top-K SAE on activations from the middle of the network and examined the resulting 1024 features along three axes: whether individual features have a clear, human-readable interpretation; whether that interpretation is reflected in the model's behaviour under feature clamping; and whether features with related interpretations sit near one another in the SAE's decoder space.

We found clear evidence that at least some features are genuinely interpretable. Features such as Black Knights (feature 426), Open Space in the Endgame (feature 253), and White Position (feature 188) show consistent, human-readable activation patterns across many positions, and in some cases appear to be sensitive to the phase of the game without being told the phase explicitly. Scaling this analysis with an LLM judge across all 1024 features suggested that this kind of clean interpretability is the exception rather than the rule: roughly 79.3% of features received the lowest interpretability score, with only around 9.9% scoring as clearly interpretable. Behavioural clamping experiments on our three example features gave mixed support for the idea that these interpretations are causally load-bearing rather than merely descriptive; the Black Knights feature responded to clamping in the direction we expected, while Open Space and White Position showed weaker, harder-to-quantify effects. Finally, an analysis of the local neighbourhood around feature 188 in decoder space found little evidence that nearby features share related interpretations, suggesting that whatever structure the SAE has learned is not organised into clean, human-legible neighbourhoods, at least at the scale we trained.

Taken together, these results suggest that SAEs can extract genuinely interpretable, causally relevant features from a chess-specific transformer, but that a 0.5M-parameter dictionary at a single layer captures only a modest fraction of the model's underlying concepts cleanly. The remainder appear to be present in some entangled or partially-formed way, consistent with the superposition hypothesis, rather than being simply absent from the model's representations.

4.1 Limitations

Several limitations qualify the strength of the conclusions we can draw from this work.

Single layer, single width. We trained one SAE at one layer (the output of layer 4 of 6) and one dictionary size (1024 features, Top-$K = 32$). We cannot say from this alone whether interpretability would improve, worsen, or simply shift in character at other layers or dictionary sizes, and our claims about feature interpretability are therefore specific to this particular configuration rather than a general statement about ChessLM.

Qualitative behavioural evidence. Our clamping experiments in Section 6 rely on visual inspection of a handful of positions per feature. This is useful for building intuition and for the kind of illustrative case study we present, but it does not constitute a quantitative measurement of effect size, and it is not clear how well our three example features (Black

Knights, Open Space, White Position) represent the broader population of features in the SAE.

LLM-graded interpretability. As discussed in Section 2.1, our headline interpretability statistic (79.3% uninterpretable) comes from a single pass of a single LLM judge over four positions per feature, without a held-out human-agreement check. This is a reasonable way to scale analysis across 1024 features, but the absolute rate should be treated as indicative rather than precise, and is likely a conservative underestimate for features whose interpretation requires chess-specific tactical or strategic reasoning rather than simple piece or square identification.

Projection artefacts in the neighbourhood analysis. Our claim that the decoder space lacks local semantic structure (Section 7) rests partly on a two-dimensional t-SNE projection of a 256-dimensional space in which the underlying cosine similarities are already weak (mean ≈ 0.004). t-SNE is known to distort distances under these conditions, so it is possible that some of the apparent lack of structure is a visualisation artefact rather than a true property of the decoder geometry, and we have not yet cross-checked this with projection-free methods.

No ground truth beyond human/LLM judgement. Our interpretations of individual features: "Black Knights", "Open Space in the Endgame", and so on, are derived from visual inspection of activation overlays rather than from an independent, objective measure of chess concepts (e.g. engine evaluations or hand-coded heuristics). This leaves open the possibility that our interpretations, while plausible, do not fully or precisely characterise what a given feature is actually computing.

Correlational, not causal, feature-function link. Even where clamping experiments show behavioural effects consistent with our interpretation, we have not established that these features are functionally necessary for ChessLM's downstream task (producing useful position embeddings) as opposed to being incidentally present but unused, or redundant with other features.

4.2 Future Work

We see several natural directions for extending this work, which we group into four themes.

Making the behavioural analysis quantitative. The clamping experiments in Section 6 could be converted from illustrative case studies into a measured effect size by running clamping across many (feature, position) pairs rather than a single example per feature, and by replacing visual judgement with cheap, board-derived ground-truth metrics: piece-count and material difference between search results, square-overlap or Hamming distance between piece placements, and concept-specific scores such as the number of empty squares for the Open Space feature or simple king-safety and pawn-structure heuristics for other features. Reporting how these metrics move as a function of clamp strength, with error bars across many sampled positions, would let us state a measured effect size rather than "limited evidence" for whether a feature's interpretation matches its causal role.

External grounding against chess concepts. A complementary direction is to correlate feature activations directly against an external source of chess knowledge – for example, a chess engine such as Stockfish, or simpler heuristics for material balance, king safety, pawn structure (doubled, isolated, and passed pawns), piece mobility, and game

phase. This would provide validation independent of our own visual judgement, and would let us test more rigorously the game-phase-invariance and endgame-specificity claims we make about individual features.

Revisiting the neighbourhood analysis. Given the limitation noted above regarding t-SNE distortion at low similarity values, a natural next step is to re-examine feature neighbourhoods without relying on a single 2D projection. This could include cross-checking with UMAP, which has different distortion characteristics to t-SNE; reporting clustering results directly in the high-dimensional decoder space via k-means or hierarchical clustering with silhouette scores, avoiding projection entirely; and running a permutation test that compares mean intra-label cosine similarity for held-out labelled features (chosen without reference to their neighbours) against a random baseline, to test directly whether same-theme features cluster more tightly than chance.

Scaling across layers and dictionary width. We trained a single SAE at a single layer and width. Training SAEs at each of ChessLM's six layers would let us test whether features become more abstract with depth – moving from low-level piece detectors in early layers toward higher-level strategic concepts in later layers – mirroring findings from the circuits and universality literature in language models. Separately, sweeping SAE width (or Top-K) at a fixed layer would let us test for feature splitting directly: whether wider dictionaries decompose coarse features such as White Position into more specific sub-features (e.g. White king safety, White pawn majority, White piece activity), as has been observed in larger-scale SAE work, and whether this splitting is accompanied by tighter, more interpretable neighbourhood structure than we observed here.

Connecting interpretability to downstream function. Finally, since ChessLM was trained to produce embeddings for downstream retrieval, the strongest test of whether a feature is genuinely load-bearing – rather than merely correlated with a chess concept – would be to ablate individual interpretable features and measure the effect on ChessLM's actual downstream embedding quality metrics. This would distinguish features the model actively relies on, such as a hypothetical "Black Knights" feature, from features that are visible in activation space but incidental to the model's function, and would connect the interpretability results in this paper directly to the practical performance of the model they describe.

References

[1] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023, https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[2] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[3] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones *et al.*, "Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet," *arXiv preprint arXiv:2605.29358*, 2026.

[4] B. Hull, "Beyond evaluation: Learning contextual chess position representations," Accessed via https://bluehood.github.io/, 2025, technical report.

[5] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 2013, pp. 746–751.

[6] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski, "Linear algebraic structure of word senses, with applications to polysemy," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 483–495, 2018.