Measuring Progress in Dictionary Learning for Language Model Interpretability with Board Game Models

Adam Karvonen [*1] **Benjamin Wright** [*2] **Can Rager** [1] **Rico Angell** [3] **Jannik Brinkmann** [4] **Logan Smith** [1]
Claudio Mayrink Verdun [5] **David Bau** [6] **Samuel Marks** [6]

Abstract

What latent features are encoded in language model (LM) representations? Recent work on training sparse autoencoders (SAEs) to disentangle interpretable features in LM representations has shown significant promise. However, evaluating the quality of these SAEs is difficult because we lack a ground-truth collection of interpretable features that we expect good SAEs to recover. We thus propose to measure progress in interpretable dictionary learning by working in the setting of LMs trained on chess and Othello transcripts. These settings carry natural collections of interpretable features—for example, "there is a knight on F3"—which we leverage into *supervised* metrics for SAE quality. To guide progress in interpretable dictionary learning, we introduce a new SAE training technique, *p-annealing*, which improves performance on prior unsupervised metrics as well as our new metrics.[1]

1. Introduction

Mechanistic interpretability aims to reverse engineer neural networks into human-understandable components. What, however, should these components be? Recent work has applied Sparse Autoencoders (SAEs) (Bricken et al., 2023; Cunningham et al., 2023), a scalable unsupervised learning method inspired by sparse dictionary learning to find a disentangled representation of language model (LM) internals. However, measuring progress in training SAEs is challenging because we do not know what a gold-standard dictionary would look like, as it is difficult to anticipate which ground-truth features underlie model cognition. Prior work has either attempted to measure SAE quality in toy synthetic settings (Sharkey et al., 2023) or relied on various proxies such as sparsity, fidelity of the reconstruction, and LM-assisted autointerpretability (Bills et al., 2023).

In this work, we explore a setting that lies between toy synthetic data (where all ground-truth features are known; cf. Elhage et al. (2022)) and natural language: LMs trained on board game transcripts. This setting allows us to formally specify natural categories of interpretable features, e.g., "there is a knight on `e3`" or "the bishop on `f5` is pinned." We leverage this to introduce two novel metrics for how much of a model's knowledge an SAEs has captured:

- **Board reconstruction.** Can we reconstruct the state of the game board by interpreting each feature as a classifier for some board configuration?
- **Coverage.** Out of a catalog of researcher-specified candidate features, how many of these candidate features actually appear in the SAE?

These metrics carry the limitation that they are sensitive to researcher preconceptions. Nevertheless, we show that they provide a useful new signal of SAE quality.

Additionally, we introduce *p-annealing*, a novel technique for training SAEs. When training an SAE with p-annealing, we use an L_p-norm-based sparsity penalty with p ranging from $p = 1$ at the beginning of training (corresponding to a convex minimization problem) to some $p < 1$ (a non-convex objective) by the end of training. We demonstrate that p-annealing improves over prior methods, giving performance on par with the more compute-intensive Gated SAEs from Rajamanoharan et al. (2024), as measured both by prior metrics and our new metrics.

Overall, our main contributions are as follows:

1. We train and open-source over 500 SAEs trained on chess and Othello models each.
2. We introduce two new metrics for measuring the quality of SAEs.

[*]Equal contribution [1]Independent [2]MIT [3]University of Massachusetts, Amherst [4]University of Mannheim [5]Harvard University [6]Northeastern University. Correspondence to: Adam Karvonen <adam.karvonen@gmail.com>.

Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Code, models, and data are available at https://github.com/adamkarvonen/SAE_BoardGameEval

3. We introduce p-annealing, a novel technique for training SAEs that improves on prior techniques.

2. Background

2.1. Language models for Othello and chess

In this work, we make use of LMs trained to autoregressively predict transcripts of chess and Othello games. We emphasize that these transcripts only give lists of moves in a standard notation and do not directly expose the board state. Based on behavioral evidence (the high accuracy of the LMs for predicting legal moves) and prior studies of LM representations (Li et al., 2023a; Nanda et al., 2023; Karvonen, 2024) we infer that the LMs internally model the board state, making them a good testbed setting for studying LM representations.

Othello. Othello is a two-player strategy board game played on an 8x8 grid, with players using black and white discs. Players take turns placing discs on the board, capturing their opponent's discs by bracketing them between their own, causing the captured discs to turn their color. The goal is to have more discs turned to display your color at the end of the game. The game ends if every square on the board is covered or either player cannot make a move.

In our experiments, we use an 8-layer GPT model with 8 attention heads and a $n = 512$ dimensional hidden space, as provided by Li et al. (2023a). This model had no prior knowledge of the game or its rules and was trained from scratch on 20 million game transcripts, where each token in the corpus represents a tile on which players place their discs. The game transcripts were synthetically generated by uniformly sampling from the Othello game tree. Thus, the data distribution captures valid move sequences rather than strategic depth. For this model, Li et al. (2023a) demonstrated the emergence of a world model—an internal representation of the correct board state allowing it to predict the next move—that can be extracted from the model activations using a non-linear probe. Nanda et al. (2023) extended this finding, showing that a similar internal representation could be extracted using linear probes, supporting the linear representation hypothesis (Mikolov et al., 2013).

Chess. Othello makes a natural testbed for studying emergent internal representations since the game tree is far too large to memorize. However, the rules and state are not particularly complex. Therefore, we also consider a language model trained on chess game transcripts with identical architecture, provided by Karvonen (2024). The model again had no prior knowledge of chess and was trained from scratch on 16 million human games from the Lichess chess game database (Lichess, 2024). The input to the model is a string in the Portable Game Notation (PGN) format (e.g., "`1. e4 e5 2. Nf3 ...`"). The model predicts a legal move in 99.8 % of cases and, similar to Othello, it has an internal representation of the board state that can be extracted from the internal activations using a linear probe (Karvonen, 2024).

2.2. Sparse autoencoders

Given a dataset $\mathcal{D}$ of vectors $\mathbf{x} \in \mathbb{R}^d$, a sparse autoencoder (SAE) is trained to produce an approximation

$$\mathbf{x} \approx \sum_{i}^{d_{\text{SAE}}} f_i(\mathbf{x})\mathbf{d}_i + \mathbf{b} \tag{1}$$

as a sparse linear combination of *features*. Here, the *feature vectors* $\mathbf{d}_i \in \mathbb{R}^d$ are unit vectors, the *feature activations* $f_i(\mathbf{x}) \geq 0$ are a sparse set of coefficients, and $\mathbf{b} \in \mathbb{R}^d$ is a bias term. Concretely, an SAE is a neural network with an encoder-decoder architecture, where the encoder maps $\mathbf{x}$ to the vector $\mathbf{f} = \begin{bmatrix} f_1(\mathbf{x}) & \dots & f_{d_{\text{SAE}}}(\mathbf{x}) \end{bmatrix}$ of feature activations, and the decoder maps $\mathbf{f}$ to an approximate reconstruction of $\mathbf{x}$.

In this paper, we train SAEs on datasets consisting of activations extracted from the residual stream after the sixth layer for both the chess and Othello models. At these layers, linear probes trained with logistic regression were accurate for classifying a variety of properties of the game board (Karvonen, 2024; Nanda et al., 2023). For training SAEs, we employ a variety of SAE architectures and training algorithms, as detailed in Section 4.

3. Measuring autoencoder quality for chess and Othello models

Many of the features learned by our SAEs reflect uninteresting, surface-level properties of the input, such as the presence of certain tokens. However, upon inspection, we additionally find many SAE features which seem to reflect a latent model of the board state, e.g., features that reflect the presence of certain pieces on particular squares, squares that are legal to play on, and strategy-relevant properties like the presence of a pin in chess (Figure 1).

Fortunately, in the setting of board games, we can formally specify certain classes of these interesting features, allowing us to more rigorously detect them and use them to understand our SAEs. In Section 3.1, we specify certain classes of interesting game board properties. Then, in Section 3.2, we leverage these classes into two metrics of SAE quality.

3.1. Board state properties in chess and Othello models

We formalize a *board state property* (BSP) to be a function $g : \{\text{game board}\} \to \{0, 1\}$. In this work, we will consider the following interpretable classes of BSPs:

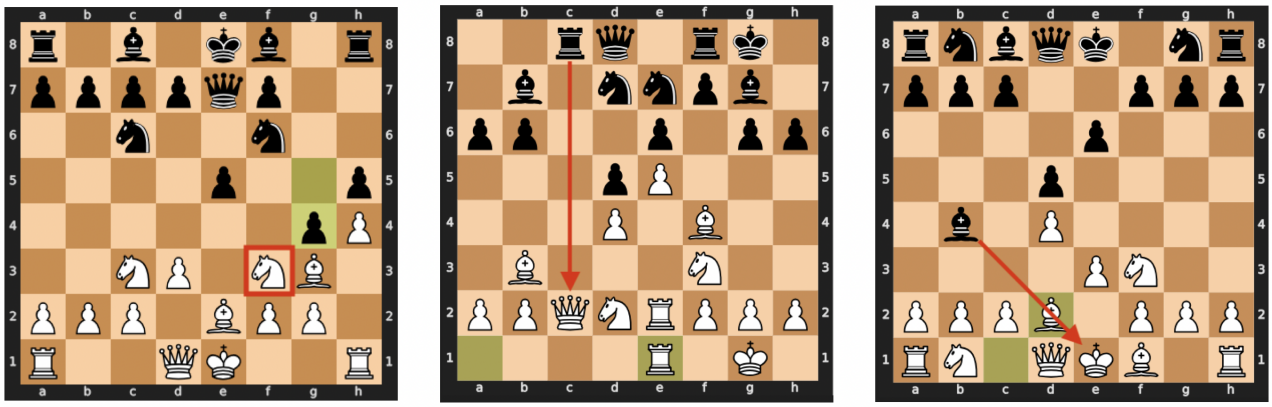


Figure 1: We find SAE features that detect interpretable board state properties (BSP) with high precision (i.e., above 0.95). This figure illustrates three distinct chessboard states, each an example of a BSP associated with a high activation of a particular SAE feature. Left: A **board state detector** identifies a knight on square `f3`, owned by the player to move. Middle: A **rook threat detector** indicates an immediate threat posed by a rook to a queen regardless of location and piece threatened. Right: A **pin detector** recognizes moves that resolve a check on a diagonal by creating a pin, again, regardless of location and piece pinned.

- $\mathcal{G}_{\text{board state}}$ contains BSPs which classify the presence of a piece at a specific board square, where the board consists of 8×8 squares in both games. For chess, we consider the full board for the twelve distinct piece types (white king, white queen, ..., black king), giving a total of $8 \times 8 \times 12$ BSPs. For Othello, we consider the full board for the two distinct piece types (black and white), yielding $8 \times 8 \times 2$ BSPs.

- $\mathcal{G}_{\text{strategy}}$ consists of BSPs relevant for predicting legal moves and playing strategically in chess, such as a pin detector. They were selected by the authors based on domain knowledge and prior interpretability work on the chess model AlphaZero (McGrath et al., 2022). We provide a full list of strategy BSPs in Table 3 in the Appendix. Because our Othello model was trained to play random legal moves, we do not consider strategy BSPs for Othello.

3.2. Measuring SAE quality with board state properties

In this section, we introduce two metrics of SAE quality: coverage and board reconstruction.

Coverage. Given a collection $\mathcal{G}$ of BSPs, our coverage metric quantifies the extent to which an SAE has identified features that coincide with the BSPs in $\mathcal{G}$. In more detail, suppose that f_i is an SAE feature and $t \in [0, 1]$ is a threshold, we define the function

$$\phi_{f_i,t}(\mathbf{x}) = \mathbb{I}\left[f_i(\mathbf{x}) > t \cdot f_i^{\max}\right] \tag{2}$$

where $f_i^{\max}$ is (an empirical estimate of) $\max_{\mathbf{x}\sim\mathcal{D}} f_i(\mathbf{x})$, the maximum value that f_i takes over the dataset $\mathcal{D}$ of activations extracted from our model, and $\mathbb{I}$ is the indicator function. We interpret $\phi_{f_i,t}$ as a binary classifier; intuitively, it corresponds to binarizing the activations of f_i into "on" vs. "off" at some fraction t of the maximum value of f_i on $\mathcal{D}$. Given some BSP $g \in \mathcal{G}$, let $F_1(\phi_{f_i,t}; g) \in [0, 1]$ denote the F1-score for $\phi_{f_i,t}$ classifying g. Then we define the *coverage* of an SAE with features $\{f_i\}$ relative to a set of BSPs $\mathcal{G}$ to be

$$\text{Cov}(\{f_i\}, \mathcal{G}) := \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \max_t \max_{f_i} F_1(\phi_{f_i,t}; g). \tag{3}$$

In other words, we take, for each $g \in \mathcal{G}$, the F_1-score of the feature that best serves as a classifier for g, and then take the mean of these maximal F_1-scores. An SAE receives a coverage score of 1 if, for each BSP $g \in \mathcal{G}$, it has some feature that is a perfect classifier for g. Since Cov depends on the choice of threshold t, we sweep over $t \in \{0, 0.1, 0.2, \ldots, 0.9\}$ and take the best coverage score; typically this best t is in $\{0, 0.1, 0.2\}$.

Board reconstruction. Again, let $\mathcal{G}$ be a set of BSPs. Intuitively, the idea of our board reconstruction metric is that, for a sufficiently good SAE, there should be a simple, human-interpretable way to recover the state of the board from the profile of feature activations $\{f_i(\mathbf{x})\}$ on an activation $\mathbf{x} \in \mathbb{R}^d$. Here, the activation $\mathbf{x}$ was extracted after the post-MLP residual connection in layer 6.

We will base our board reconstruction metric around the following human-interpretable way of recovering a board state from a feature activation profile; we emphasize that different ways of recovering boards from feature activations

may lead to qualitatively different results. This recovery rule is based on the assumption that interpretable SAE features tend to be *high precision* for some subset of BSPs, in line with Templeton et al. (2024). For example, features that classify common configurations of pieces are high precision (but not necessarily high recall) for multiple BSPs. Thus, using a batch of "train" data $\mathcal{D}_{\text{train}}$ we identify, for each SAE feature f_i, all of the BSPs $g \in \mathcal{G}$ for which $\phi_{f_i,t}$ is a *high precision* (of at least 0.95) classifier. Then for each $g \in \mathcal{G}$ our prediction rule is

$$\mathcal{P}_g(\{f_i(\mathbf{x})\}) = \begin{cases} 1, & \text{if } \phi_{f_i,t}(\mathbf{x}) = 1 \text{ for any } f_i \text{ which} \\ & \quad \text{is high precision for } g \text{ on } \mathcal{D}_{\text{train}} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Let $F_1(\boldsymbol{\mathcal{P}}(\{f_i(\mathbf{x})\}); \mathbf{b})$ denote the F_1-score for a given board state $\mathbf{b}$, where $\boldsymbol{\mathcal{P}}(\{f_i(\mathbf{x})\}) = \{\mathcal{P}_g(\{f_i(\mathbf{x})\})\}_{g \in \mathcal{G}}$ represents the full predicted board (containing predictions for all 64 squares) obtained from the SAE activations. Then, the average F_1-score over all board states in the test dataset $\mathcal{D}_{\text{test}}$ can be calculated as:

$$\text{Rec}(\{x_i\}, \mathcal{D}_{\text{test}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{\mathbf{x} \in \mathcal{D}_{\text{test}}} \max_t F_1(\boldsymbol{\mathcal{P}}(\{f_i(\mathbf{x})\}); \mathbf{b}), \tag{5}$$

4. Training methodologies for SAEs

In our experiments, we investigate four methods for training SAEs, as explained in this section. These are given by two autoencoder architectures and two training methodologies—one with p-annealing and one without p-annealing—for each architecture. Our SAEs are available at `https://huggingface.co/adamkarvonen/chess_saes/tree/main` (chess) and `https://huggingface.co/adamkarvonen/othello_saes/tree/main` (Othello).

4.1. Standard SAEs

Let n be the dimension of the model's residual stream activations that are input to the autoencoder, m the autoencoder hidden dimension, and s the dataset size. Our baseline "standard" SAE architecture, as introduced in Bricken et al. (2023) is defined by encoder weights $W_e \in \mathbb{R}^{m \times n}$, decoder weights $W_d \in \mathbb{R}^{n \times m}$ with columns constrained to have a L_2-norm of 1, and biases $b_e \in \mathbb{R}^m$, $b_d \in \mathbb{R}^n$. Given an input $\mathbf{x} \in \mathbb{R}^n$, the SAE computes

$$\mathbf{f}(\mathbf{x}) = \text{ReLU}(W_e(\mathbf{x} - \mathbf{b}_d) + \mathbf{b}_e) \tag{6}$$

$$\hat{\mathbf{x}} = W_d\, \mathbf{f}(\mathbf{x}) + \mathbf{b}_d \tag{7}$$

where $\mathbf{f}(\mathbf{x})$ is the vector of feature activations, and $\hat{\mathbf{x}}$ is the reconstruction.

For a standard SAE, our baseline training method is as implemented in the open-source `dictionary_learning` repository (Marks & Mueller, 2024), optimizing the loss

$$\mathcal{L}_{\text{standard}} = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}} \Big[\|\mathbf{x} - \hat{\mathbf{x}}\|_2 + \lambda \|\mathbf{f}(\mathbf{x})\|_1 \Big]. \tag{8}$$

for some hyperparameter $\lambda > 0$ controlling sparsity.

4.2. Gated SAEs

The L_1 penalty used in the original training method encourages feature activations to be smaller than they would be for optimal reconstruction (Wright & Sharkey, 2024). To address this, Rajamanoharan et al. (2024) introduced a modification to the original SAE architecture that separates the selection of dictionary elements to use in a reconstruction and estimating the coefficients of these dictionary elements. This results in the following gated architecture:

$$\pi_{\text{gate}}(\mathbf{x}) := W_{\text{gate}}(\mathbf{x} - \mathbf{b}_d) + \mathbf{b}_{\text{gate}}$$

$$\tilde{\mathbf{f}}(\mathbf{x}) := \mathbb{I}\left[\pi_{\text{gate}}(\mathbf{x}) > 0\right] \odot \text{ReLU}(W_{\text{mag}}(\mathbf{x} - \mathbf{b}_d) + \mathbf{b}_{\text{mag}})$$

$$\hat{x}(\tilde{\mathbf{f}}(\mathbf{x})) = W_d \tilde{\mathbf{f}}(\mathbf{x}) + \mathbf{b}_d$$

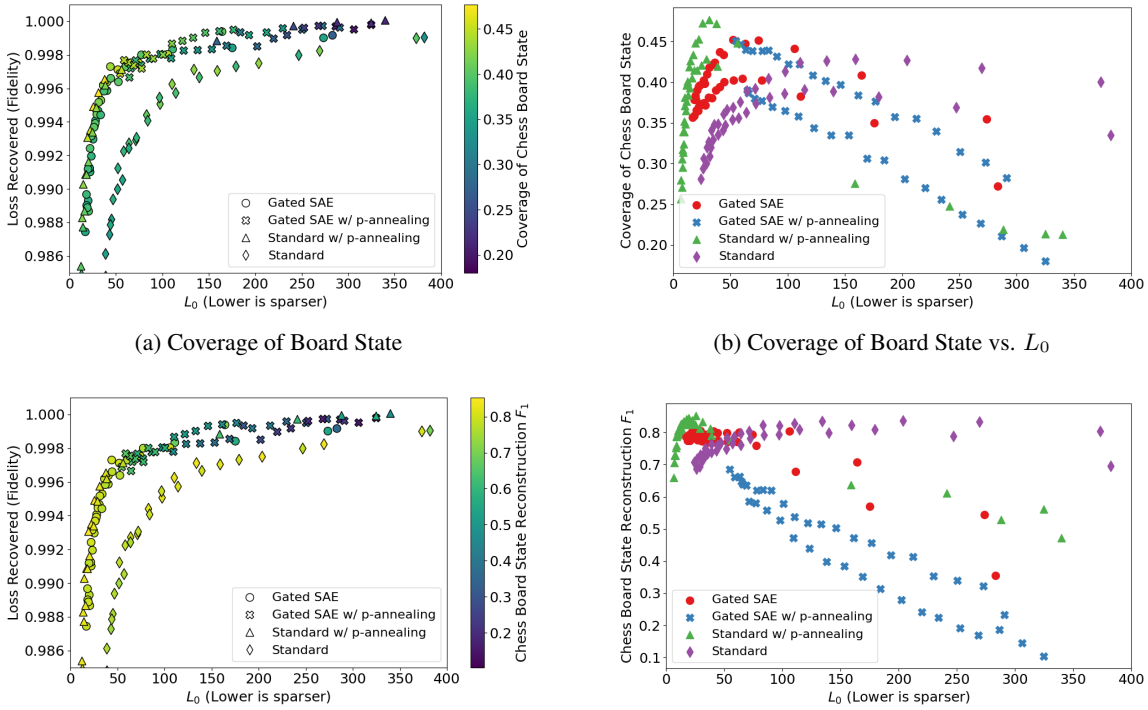
where $\mathbb{I}[\cdot > 0]$ is the Heaviside step function and $\odot$ denotes elementwise multiplication. Then, the loss function uses $\hat{x}_{\text{frozen}}$, a frozen copy of the decoder:

$$\begin{aligned} \mathcal{L}_{\text{gated}} := \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}} \Big[& \|\mathbf{x} - \hat{x}(\tilde{\mathbf{f}}(\mathbf{x}))\|_2^2 \\ & + \lambda \|\text{ReLU}(\pi_{\text{gate}}(\mathbf{x}))\|_1 \\ & + \|\mathbf{x} - \hat{x}_{\text{frozen}}(\text{ReLU}(\pi_{\text{gate}}(\mathbf{x})))\|_2^2 \Big] \end{aligned} \tag{9}$$

4.3. p-Annealing

Fundamentally, an L_1 penalty has been used to induce sparsity in SAE features because it serves as a convex relaxation of the true sparsity measure, the L_0-norm. The L_1-norm is the convex hull of the L_0-norm, making it a tractable alternative for promoting sparsity (Wright & Ma, 2022). However, the proxy loss function is not the same as directly optimizing for sparsity, leading to issues such as feature *shrinkage* (Wright & Sharkey, 2024) and potentially less sparse learned features. Unfortunately, the L_0-norm is non-differentiable and directly minimizing it is an NP-hard problem (Natarajan, 1995; Davis et al., 1997), rendering it impractical for training.

In this work, we propose the use of nonconvex L_p^p-minimization, with $p < 1$, as an alternative to the standard L_1 minimization in sparse autoencoders (SAEs). This approach has been successfully employed in compressive sensing and sparse recovery to achieve even sparser representations (Chartrand, 2007; Wen et al., 2015; Yang et al., 2018; Wang et al., 2011). To perform this optimization, we introduce a method called *p-annealing* for training SAEs, based on the compressive sensing technique called



(a) Coverage of Board State

(b) Coverage of Board State vs. L_0

(c) Board Reconstruction

(d) Board Reconstruction vs. L_0

Figure 2: Comparison of the coverage and board reconstruction metrics for chess SAE quality on $\mathcal{G}_{\text{board state}}$. The coverage score reports the mean F1 scores over BSPs. The top row corresponds to coverage, and the bottom row corresponds to board reconstruction. The left column contains a scatterplot of loss recovered vs. L_0, with the scheme color corresponding to the coverage score and each point representing different hyperparameters. We differentiate between SAE training methods with shapes.

p-continuation (Zheng et al., 2017). The key idea is to start with convex L_1-minimization through setting $p = 1$ and progressively decrease the value of p during training, resulting in closer approximations of the true sparsity measure, L_0, as p approaches 0. We define the sparsity penalty for each batch x as a function of the current training step s:

$$\mathcal{L}_{\text{sparse}}(\mathbf{x}, s) = \lambda_s \|\mathbf{f}(\mathbf{x})\|_{p_s}^{p_s} = \lambda_s \sum_i f_i(\mathbf{x})^{p_s} \tag{10}$$

In other words, the sparsity penalty will be a scaled L_p^p norm of the SAE feature activations, with p decreasing over time. At $p = 1$, the L_p^p norm is equal to the L_1 norm, and as $p \to 0$, the L_p^p norm limits to the L_0-norm, as $\lim_{p\to 0} \sum_i f_i(\mathbf{x})^p = \sum_i f_i(\mathbf{x})^0$.

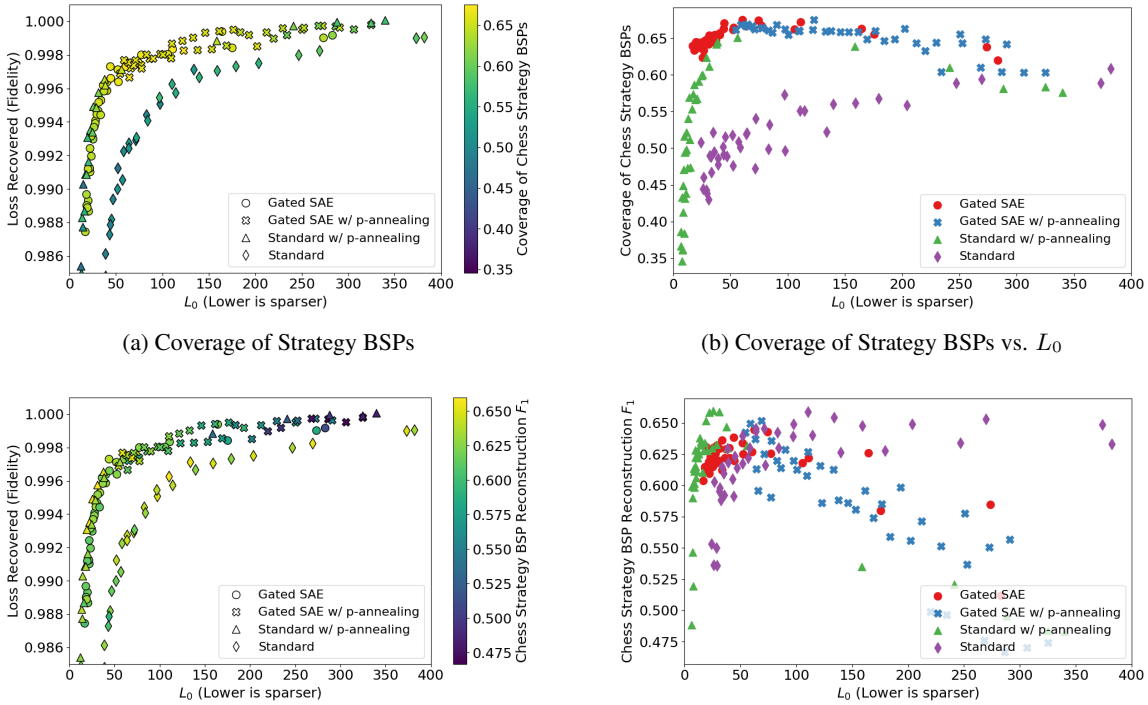
The purpose of annealing p from $1 \to 0$ instead of starting from a fixed, low value for p is that the lower the p, the more concave (non-convex) the L_p^p norm is, increasing the likelihood of the training process getting stuck in local optima, which we have observed in initial experiments. Therefore, we aim to first arrive at a region of an optimum using the easier-to-train L_1 penalty and then gradually shift the loss function. This manifests as keeping $p = 1$ for a certain number of steps and then starting decreasing p linearly down to $p_{\text{end}} > 0$ at the end of training. We set $p_{\text{end}} = 0.2$.

Coefficient Annealing. Changing the value of p changes the scale of the L_p^p norm. Without also adapting the coefficient λ, the strength of the sparsity penalty would vary too wildly across training. Empirically, we found that keeping a constant λ would lead to far too weak of a sparsity penalty for the larger p's at the start of training, making the process worse than simply training with a constant p from the beginning.

Consequently, we aim to adapt the coefficient λ such that the strength of the sparsity penalty is not changed significantly due to p updates. Formally, the update step is:

$$\lambda_{s+1} \leftarrow \lambda_s \frac{\sum_{j=s-q+1}^{s} \sum_i f_i(\mathbf{x_j})^{p_s}}{\sum_{j=s-q+1}^{s} \sum_i f_i(\mathbf{x_j})^{p_{s+1}}}. \tag{11}$$

We keep a queue of the most recent q batches of feature activations mid-training and use them to calibrate the λ_s updates. Therefore, the strength of the sparsity penalty is kept *locally* constant.



(a) Coverage of Strategy BSPs

(b) Coverage of Strategy BSPs vs. L_0

(c) Strategy BSP Reconstruction

(d) Strategy BSP Reconstruction vs. L_0

Figure 3: Comparison of the coverage and board reconstruction metrics for chess SAE quality on $\mathcal{G}_{\text{strategy}}$. The metrics represent the average coverage and board reconstruction obtained across all BSPs in $\mathcal{G}_{\text{strategy}}$. The coverage score reports the mean of maximal F1 scores over BSPs. The absolute coverage scores vary significantly between strategy BSPs, as discussed in Appendix C. The top row corresponds to coverage, and the bottom row corresponds to board reconstruction. The left column contains a scatterplot of loss recovered vs. L_0, with the color scheme corresponding to the coverage score and each point representing different hyperparameters. We differentiate between SAE training methods with shapes.

Combining p-annealing with other SAEs. Since the p-annealing method only modifies the L_1 terms in the loss function without affecting the SAE architecture, it is simple to combine p-annealing with other SAE modifications. This allows us to create the Gated-Annealed SAE method by combining the Gated SAE architecture and p-annealing. Concretely, we modify $\mathcal{L}_{\text{gated}}$ (Equation 9) by replacing the sparsity term $\lambda\|\text{ReLU}(\pi_{\text{gate}}(\mathbf{x}))\|_1$ in with $\lambda_s\|\text{ReLU}(\pi_{\text{gate}}(\mathbf{x}))\|_{p_s}^{p_s}$. Our experiments showed that the optimum values for coefficients λ and λ_s differ.

5. Results

In this section, we explore the performance of SAEs applied to language models trained on Othello and chess. Consistent with Nanda et al. (2023), we find that interpretable SAE features typically track properties relative to the player whose turn it is (e.g. "my king is pinned" rather than "the white king is pinned"). To side-step subtleties arising from this, we only extract our activations from the token immediately preceding white's move. Specifically, we consider SAEs trained on the residual stream activations after the sixth layer using the four methods from Section 4 (see Table 2 for additional hyperparameters). In addition to our metrics introduced above, we also make use of unsupervised metrics previously appearing in the literature (Bricken et al., 2023; Cunningham et al., 2023; Rajamanoharan et al., 2024):

- $\boldsymbol{L_0}$ measures the average number of active SAE active features (i.e., positive activation) on a given input.
- **Loss recovered** measures the change in model performance when replacing activations with the corresponding SAE reconstruction during a forward pass. This metric is quantified as $(H_* - H_0)/(H_{\text{orig}} - H_0)$, where H_{orig} is the cross-entropy loss of the board game model for next-token prediction, H_* is the cross-entropy loss after substituting the model activation $\mathbf{x}$ with its SAE reconstruction during the forward pass, and H_0 is the cross-entropy loss when zero-ablating $\mathbf{x}$.

Our key takeaways are as follows.

SAE features can accurately reconstruct game boards. In general, we find that SAE features are effective at captur-

	Chess		Othello	
Model	**Coverage**	**Reconstruction**	**Coverage**	**Reconstruction**
SAE: random GPT	0.11	0.01	0.27	0.08
SAE: trained GPT	0.48	0.85	0.52	0.95
Linear probe	0.98	0.98	0.99	0.99

Table 1: Best performance obtained for different techniques across games for $\mathcal{G}_{\text{board state}}$. As a baseline, we train an SAE on random GPT, a version of the trained GPT model with randomly initialized weights. All models were trained on activations after the post-MLP residual connection in layer 6.

ing board state information in both Othello and chess (see Table 1, Figure 2d and 4d). In contrast, SAEs trained on a model with random weights perform very poorly according to our metrics, showing that SAE performance is driven by identifying structure in the models' learned representation of game boards. Nonetheless, SAEs do not match the performance of linear probes in terms of reconstructing the board state. This performance gap suggests that SAEs do not capture all of the information encoded in the model's internal representations.

Standard SAEs trained with p-annealing perform on par with Gated SAEs. We find that standard SAEs trained using p-annealing consistently perform better than those trained with a constant L_1 penalty (Equation 8), as measured by existing proxy metrics and in terms of improvement in coverage (see Figure 2a and 4a). In fact, standard SAEs trained using p-annealing show a coverage score that is comparable to Gated SAEs trained without p-annealing. However, we find cases where our coverage metric disagrees with existing metrics. In Figure 2, for example, Gated SAEs perform achieve a higher *loss recovered* score than Standard SAEs trained using p-annealing. We emphasize that the training and inference of Gated SAEs is more computationally expensive, requiring 50% more compute per forward pass compared to Standard SAEs (Rajamanoharan et al., 2024).

Coverage and board reconstruction reveal differences in SAE quality not captured by unsupervised metrics. Our metrics reveal improvements in SAE performance that traditional proxy metrics fail to capture. For example, we trained SAEs with hidden dimensions 4096 and 8192 (expansion factors of 8 and 16, respectively). We expect the SAEs with 8192 hidden dimensions to perform better since they have greater capacity. However, we observe that they perform equally well according to prior unsupervised metrics (see Figures 2 a, c and 4 a, c). In contrast, our metrics reveal that SAEs with larger hidden dimensions are better. For the Standard architecture, this is reflected by the parallel lines (of purple diamonds) in Figures 2 b, d and 4 b, d. Thus, our metrics are able to capture improvements from larger expansion factors. In addition, we find that the performance of p-annealing closely resembles that of Gated SAEs when evaluated using standard proxy metrics; it demonstrates clear improvements under our proposed metrics.

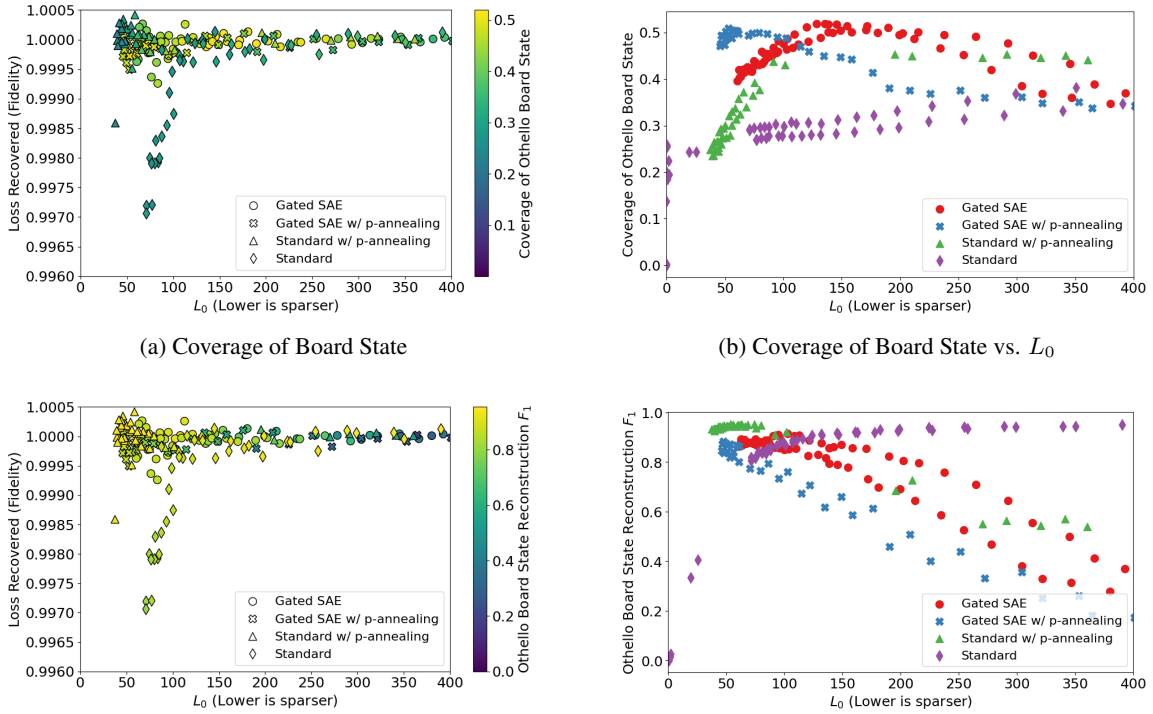
Coverage and board reconstruction are consistent with existing metrics. Figures 2, 4, and 3 demonstrate that both coverage and board reconstruction metrics are optimal in the elbow region of the Pareto frontier. This region, where SAEs reconstruct internal activations efficiently with minimal features, also yielded the most coherent interpretations during our manual inspections. This provides precise, empirical validation to the common wisdom that SAEs in this region of the Pareto frontier are the best.

6. Limitations

The proposed metrics for board reconstruction and coverage provide a more objective evaluation of SAE quality than previous subjective methods. Nevertheless, these metrics exhibit several limitations. Primarily, their applicability is confined to the chess and Othello domains, raising concerns about their generalizability to other domains or different models. Additionally, the set of BSPs that underpin these metrics is determined by researchers based on their domain knowledge. This approach may not encompass all pertinent features or strategic concepts, thus potentially overlooking essential aspects of model evaluation. Developing comparable objective metrics for other domains, such as natural language processing, remains a significant challenge. Moreover, our current focus is on evaluating the quality of SAEs in terms of their ability to capture internal representations of the model. However, this does not directly address how these learned features could be utilized for downstream interpretability tasks.

7. Related work

Sparse dictionary learning. Since the nineties, dictionary learning (Elad, 2010; Dumitrescu & Irofti, 2018), sparse regression (Foucart & Rauhut, 2013), and later, sparse autoencoders (Ng et al., 2011) have been extensively studied



(a) Coverage of Board State

(b) Coverage of Board State vs. L_0

(c) Board Reconstruction

(d) Board Reconstruction vs. L_0

Figure 4: Comparison of the coverage and board reconstruction metrics for Othello SAE quality on $\mathcal{G}_{\text{board state}}$. The coverage score reports the mean of maximal F1 scores over BSPs. The top row corresponds to coverage, and the bottom row corresponds to board reconstruction. The left column contains a scatterplot of loss recovered vs. L_0, with the color scheme corresponding to the coverage score and each point representing different hyperparameters. We differentiate between SAE training methods with shapes.

in the machine learning and signal processing literature. The seminal work of Olshausen & Field (1996) introduced the concept of sparse coding in neuroscience (see also (Olshausen & Field, 2004), building upon the earlier concept of sparse representations (Donoho, 1992) and matching pursuit (Mallat & Zhang, 1993). Subsequently, a series of works established the theoretical and algorithmic foundations of sparse dictionary learning (Elad & Bruckstein, 2002; Hoyer, 2002; Eggert & Korner, 2004; Aharon et al., 2006; Yang et al., 2009; Jung et al., 2014; Tillmann, 2014; Arora et al., 2015; Bao et al., 2015; Blasiok & Nelson, 2016; Chalk et al., 2018). Notably, Gregor & LeCun (2010) introduced LISTA, an unrolled version of ISTA (Daubechies et al., 2004) that learns the dictionary instead of having it fixed.

In parallel, autoencoders were introduced in machine learning to automatically learn data features and perform dimensionality reduction (Hinton & Zemel, 1993; Li et al., 2023b). Inspired by sparse dictionary learning, sparse autoencoders (Ng et al., 2011; Coates et al., 2011; Coates & Ng, 2011; Makhzani & Frey, 2013; Li et al., 2016) were proposed as an unsupervised learning model to build deep sparse hierarchical models of data, assuming a certain degree of sparsity in the hidden layer activations. Later, Luo et al. (2017) generalized sparse autoencoders (SAEs) to convolutional SAEs. Although the theory for SAEs is less developed than that of dictionary learning with a fixed dictionary, some progress has been made in quantifying whether autoencoders can, indeed, do sparse coding, e.g., Arpit et al. (2016); Rangamani et al. (2018); Nguyen et al. (2019).

Feature disentanglement using sparse autoencoders. The individual computational units of neural networks are often polysemantic, i.e., they respond to multiple seemingly unrelated inputs (Arora et al., 2018). Elhage et al. (2022) investigated this phenomenon and suggested that neural networks represent features in linear superposition, which allows them to represent more features than they have dimensions. Thus, in an internal representation of dimension n, a model can encode $m \gg n$ concepts as linear directions (Park et al., 2023), such that only a sparse subset of concepts are active across all inputs – a concept deeply related to the coherence of vectors (Foucart & Rauhut, 2013) and to frame theory in general (Christensen et al., 2003). To identify these concepts, Sharkey et al. (2023) used SAEs to perform dictionary learning on a one-layer transformer,

identifying a large (overcomplete) basis of features. Cunningham et al. (2023) applied SAEs to language models and demonstrated that dictionary features can be used to localize and edit model behavior. Marks et al. (2024) proposed a scalable method to discover sparse feature circuits, as opposed to circuits consisting of polysemantic model components, and demonstrated that a human could change the generalization of a classifier by editing its feature circuit. Recently, Kissane et al. (2024) explored autoencoders for attention layer outputs. These works have benefited from a variety of open-source libraries for training SAEs for LLM interpretability (Marks & Mueller, 2024; Bloom & Chanin, 2024; Cooney, 2023).

Improving and evaluating sparse autoencoders. Despite the success of SAEs at extracting human-interpretable features, they fail to perfectly reconstruct the activations (Cunningham et al., 2023). One challenge in the training of SAEs with an L_1 penalty is shrinkage (or 'feature suppression'); in addition to encouraging sparsity, an L_1 penalty encourages feature activations to be smaller than they would be otherwise. Wright & Sharkey (2024) approached this problem by fine-tuning the sparse autoencoder without a sparsity penalty. Jermyn et al. (2024) and Riggs & Brinkmann (2024) explored alternative sparsity penalties to reduce feature suppression during training. Rajamanoharan et al. (2024) introduced Gated SAEs, an architectural variation for the encoder which both addresses shrinkage and improves on the Pareto frontier of L_0 vs reconstruction error. Recently, Gao et al. (2024) systematically evaluated the scaling laws with respect to sparsity, autoencoder size, and language model size.

The goal of dictionary learning in machine learning is to produce human-interpretable features and capture the underlying model's computations (Bricken et al., 2023). However, quantitatively measuring interpretability is difficult and often involves manual inspection. Therefore, most existing work assesses the quality of SAEs along different proxy metrics: (1) The cross-entropy loss recovered, which reflects the degree to which the original loss of the language model can be recovered when replacing activations with the autoencoder predictions. (2) The L_0-norm of feature activations $\mathbb{E}_{z\sim\mathcal{D}} \|h(z)\|_0$, measuring the number of activate features given an input (Ferrando et al., 2024). Makelov et al. (2024) proposed to compare SAEs against supervised feature dictionaries in a natural language setting. However, this requires a significant understanding of the model's internal computations and is thus not scalable.

8. Conclusion

Most SAE research has relied on proxy metrics such as loss recovered and L_0, or subjective manual evaluation of interpretability by examining top activations. However, proxy measures only serve as an estimate of interpretability, monosemantic nature, and comprehensiveness of the learned features, while manual evaluations depend on the researcher's domain knowledge and tend to be inconsistent.

Our work provides a new, more objective paradigm for evaluating the quality of an SAE methodology; coverage serves as a quantifiable measure of monosemanticity and quality of feature extraction, while board reconstruction serves as a quantifiable measure of the extent to which an SAE is exhaustively representing the information contained within the language model. Therefore, the optimal SAE methodology can be judged by whether it yields both high coverage and high board reconstruction.

Finally, we propose the p-annealing method, a modification to the SAE training paradigm that can be combined with other SAE methodologies and results in an improvement in both coverage and board reconstruction.

Author Contributions

A.K. built and maintained our infrastructure for working with board-game models. A.K., S.M., C.R., J.B., and L.S. designed the proposed metrics. B.W. performed initial experiments demonstrating the benefits of training SAEs with $p < 1$. B.W., C.M.V., and S.M. then proposed p-annealing, with B.W. leading the implementation and developing coefficient annealing. The basic framework for our dictionary learning work was built and maintained by S.M. and C.R. The training algorithms studied were implemented by S.M., C.R., B.W., R.A., and J.B. R.A. trained the SAEs used in our experiments. A.K., C.R., and J.B. selected and implemented the BSPs. A.K. and J.B. trained the linear probes. Many of the authors (including L.S., J.B., R.A.) did experiments applying traditional dictionary learning methods and exploring both toy problems and natural language settings, which helped build valuable intuition. The manuscript was primarily drafted by A.K., B.W., C.R., R.A., J.B., C.M.V., and S.M., with extensive feedback and editing from all authors. D.B. suggested the original project idea.

Acknowledgments

C.R. is supported by Manifund Regrants and AISST. L.R. is supported by the Long Term Future Fund. S.M. is supported by an Open Philanthropy alignment grant.

The work reported here was performed in part by the University of Massachusetts Amherst Center for Data Science and the Center for Intelligent Information Retrieval, and in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative.

References

Aharon, M., Elad, M., and Bruckstein, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

Arora, S., Ge, R., Ma, T., and Moitra, A. Simple, efficient, and neural algorithms for sparse coding. In *Conference on learning theory*, pp. 113–149. PMLR, 2015.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018. doi: 10.1162/tacl_a_00034. URL https://aclanthology.org/Q18-1034.

Arpit, D., Zhou, Y., Ngo, H., and Govindaraju, V. Why regularized auto-encoders learn sparse representation? In *International Conference on Machine Learning*, pp. 136–144. PMLR, 2016.

Bao, C., Ji, H., Quan, Y., and Shen, Z. Dictionary learning for sparse coding: Algorithms and convergence analysis. *IEEE transactions on pattern analysis and machine intelligence*, 38(7):1356–1369, 2015.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2023.

Blasiok, J. and Nelson, J. An improved analysis of the erspud dictionary learning algorithm. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

Bloom, J. and Chanin, D. Sae lens. https://github.com/jbloomAus/SAELens, 2024.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, pp. 2, 2023.

Chalk, M., Marre, O., and Tkačik, G. Toward a unified theory of efficient, predictive, and sparse coding. *Proceedings of the National Academy of Sciences*, 115(1): 186–191, 2018.

Chartrand, R. Exact reconstruction of sparse signals via nonconvex minimization. *IEEE Signal Processing Letters*, 14(10):707–710, 2007.

Christensen, O. et al. *An introduction to frames and Riesz bases*, volume 7. Springer, 2003.

Coates, A. and Ng, A. Y. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 921–928, 2011.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Cooney, A. Sparse autoencoder library. https://github.com/ai-safety-foundation/sparse_autoencoder, 2023.

Cunningham, H., Ewart, A., Smith, L. R., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

Davis, G., Mallat, S., and Avellaneda, M. Adaptive greedy approximations. *Constructive approximation*, 13:57–98, 1997.

Donoho, D. L. Superresolution via sparsity constraints. *SIAM journal on mathematical analysis*, 23(5):1309–1331, 1992.

Dumitrescu, B. and Irofti, P. *Dictionary learning algorithms and applications*. Springer, 2018.

Eggert, J. and Korner, E. Sparse coding and nmf. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 4, pp. 2529–2533. IEEE, 2004.

Elad, M. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

Elad, M. and Bruckstein, A. M. A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 48(9):2558–2567, 2002.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Ferrando, J., Sarti, G., Bisazza, A., and Costa-jussà, M. R. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

Foucart, S. and Rauhut, H. *A Mathematical Introduction to Compressive Sensing*. Springer New York, New York, NY, 2013.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Gregor, K. and LeCun, Y. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pp. 399–406, 2010.

Hinton, G. E. and Zemel, R. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Hoyer, P. O. Non-negative sparse coding. In *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pp. 557–565. IEEE, 2002.

Jermyn, A., Templeton, A., Batson, J., and Bricken, T. Tanh penalty in dictionary learning, 2024. URL https://transformer-circuits.pub/2024/feb-update/index.html. Accessed: 2024-05-20.

Jung, A., Eldar, Y. C., and Görtz, N. Performance limits of dictionary learning for sparse coding. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pp. 765–769. IEEE, 2014.

Karvonen, A. Emergent world models and latent variable estimation in chess-playing language models, 2024.

Kissane, C., Krzyzanowski, R., Bloom, J. I., Conmy, A., and Nanda, N. Interpreting attention layer outputs with sparse autoencoders. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

Li, J., Zhang, T., Luo, W., Yang, J., Yuan, X.-T., and Zhang, J. Sparseness analysis in the pretraining of deep neural networks. *IEEE transactions on neural networks and learning systems*, 28(6):1425–1438, 2016.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023a.

Li, P., Pei, Y., and Li, J. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, 138:110176, 2023b.

Lichess. lichess.org open database, 2024. URL https://database.lichess.org.

Luo, W., Li, J., Yang, J., Xu, W., and Zhang, J. Convolutional sparse autoencoders for image classification. *IEEE transactions on neural networks and learning systems*, 29 (7):3289–3294, 2017.

Makelov, A., Lange, G., and Nanda, N. Towards principled evaluations of sparse autoencoders for interpretability and control. *arXiv preprint arXiv:2405.08366*, 2024.

Makhzani, A. and Frey, B. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

Mallat, S. G. and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

Marks, S. and Mueller, A. dictionary_learning. https://github.com/saprmarks/dictionary_learning, 2024.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

McGrath, T., Kapishnikov, A., Tomašev, N., Pearce, A., Wattenberg, M., Hassabis, D., Kim, B., Paquet, U., and Kramnik, V. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119 (47), November 2022. ISSN 1091-6490. doi: 10.1073/pnas.2206625119. URL http://dx.doi.org/10.1073/pnas.2206625119.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. In Belinkov, Y., Hao, S., Jumelet, J., Kim, N., McCarthy, A., and Mohebbi, H. (eds.), *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.blackboxnlp-1.2. URL https://aclanthology.org/2023.blackboxnlp-1.2.

Natarajan, B. K. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

Ng, A. et al. Sparse autoencoder. *CS294A Lecture notes*, 72 (2011):1–19, 2011.

Nguyen, T. V., Wong, R. K., and Hegde, C. On the dynamics of gradient descent for autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2858–2867. PMLR, 2019.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Olshausen, B. A. and Field, D. J. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*, 2024.

Rangamani, A., Mukherjee, A., Basu, A., Arora, A., Ganapathi, T., Chin, S., and Tran, T. D. Sparse coding and autoencoders. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 36–40. IEEE, 2018.

Riggs, L. and Brinkmann, J. Improving sparse autoencoders by square-rooting l1 and removing lowest activation features, 2024. URL https://www.lesswrong.com/posts/YiGs8qJ8aNBgwt2YN/improving-sae-s-by-sqrt-ing-l1-and-removing-lowest. Accessed: 2024-05-20.

Sharkey, L., Braun, D., and Millidge, B. Taking features out of superposition with sparse autoencoders, 2023. URL https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition. Accessed: 2023-05-10.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Tillmann, A. M. On the computational intractability of exact and approximate dictionary learning. *IEEE Signal Processing Letters*, 22(1):45–49, 2014.

Wang, M., Xu, W., and Tang, A. On the performance of sparse recovery via ℓ_p-minimization $(0 \leq p \leq 1)$. *IEEE Transactions on Information Theory*, 57(11):7255–7278, 2011.

Wen, J., Li, D., and Zhu, F. Stable recovery of sparse signals via lp-minimization. *Applied and Computational Harmonic Analysis*, 38(1):161–176, 2015.

Wright, B. and Sharkey, L. Addressing feature suppression in sparse autoencoders, 2024. URL https://www.lesswrong.com/posts/3JuSjTZyMzaSeTxKk/addressing-feature-suppression-in-saes. Accessed: 2024-05-20.

Wright, J. and Ma, Y. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press, 2022.

Yang, C., Shen, X., Ma, H., Gu, Y., and So, H. C. Sparse recovery conditions and performance bounds for ℓ_p-minimization. *IEEE Transactions on Signal Processing*, 66(19):5014–5028, 2018.

Yang, J., Yu, K., Gong, Y., and Huang, T. Linear spatial pyramid matching using sparse coding for image classification. In *2009 IEEE Conference on computer vision and pattern recognition*, pp. 1794–1801. IEEE, 2009.

Zheng, L., Maleki, A., Weng, H., Wang, X., and Long, T. Does ℓ_p-minimization outperform ℓ_1-minimization? *IEEE Transactions on Information Theory*, 63(11):6896–6935, 2017.

A. Sparse Autoencoder Training Parameters

We used a single NVIDIA A100 GPU for training SAEs and experiments. It takes much less than 24 hours to train a single SAE on 300 million tokens. Given a trained SAE, our evaluation requires less than 5 minutes of computing time.

Table 2: Training parameters of our sparse autoencoders.

Parameter	**Value**
Number of tokens	300M
Optimizer	`Adam`
Adam betas	(0.9, 0.999)
Linear warmup steps	1,000
Batch size	8,192
Learning rate	3e-4
Expansion factor	$\{8, 16\}$
Annealing start	10,000
p_{end}	0.2
λ_{init}	[0.02, 2.0]

B. List of Board State Properties

Table 3 summarizes the high-level board state properties considered in $\mathcal{G}_{\text{strategy}}$. The selection of concepts was inspired by McGrath et al. (2022). The column indicated by # denotes the number of individual BSPs per concept. A single BSP per concept indicates we match this condition globally for any corresponding piece.

Table 3: List of strategic Board State Properties.

Concept	#	Description
`check`	1	Indicates whether the player to move is checked by the opponent.
`can_check`	1	Indicates whether the player to move could check the opponent with the next move.
`queen`	1	Indicates whether the player to move has a queen on the board.
`can_capture_queen`	1	Indicates whether the player to move can capture the queen of the opponent.
`bishop_pair`	1	Indicates whether the player to move still has both bishops on the board.
`castling_rights`	1	Indicates whether the player to move is still allowed to castle, contingent on the king and the rooks not having moved.
`kingside_castling_rights`	1	Indicates whether the player to move is still allowed to kingside castle, contingent on the king and the kingside rook not having moved.
`queenside_castling_rights`	1	Indicates whether the player to move is still allowed to queenside castle, contingent on the king and the queenside rook not having moved.
`fork`	1	Indicates whether the player to move attacks has a fork on major pieces of the opponent.
`pin`	1	Indicates whether there is a pin on the board, such that a player's piece cannot move without exposing the kind behind it to capture.
`legal_en_passant`	1	Indicates whether the player to move has a legal en passant: a special pawn capture that can only occur immediately after an opponent moves a pawn two squares from its starting position and it lands beside the player's pawn.
`ambiguous_moves`	1	Indicates whether there are moves that would require further specification as more than one piece of the same type can move to the same square.
`threatened_squares`	64	Indicates which squares are threatened by the opponent.
`legal_moves`	64	Indicates which squares can be legally moved to by the current player.

C. Performance of Linear Probes and SAEs on Board State Properties

In Figure 3, we present a mean coverage score over strategy board state properties $\mathcal{G}_{\text{BSP}}$. Properties within $\mathcal{G}_{\text{BSP}}$ vary significantly in complexity. For example, queen detection can be inferred directly from the move history, while fork detection requires an accurate representation of the board state. Table 4 shows that linear probe F1-score is below 0.95 for 6 out of 15 properties in $\mathcal{G}_{\text{BSP}}$. This suggests that chess-GPT (Karvonen, 2024) does not represent these properties linearly. Additional experiments are required to determine whether the representation is present at all.

For the board state case, reconstruction is significantly higher than coverage. This is because there are many SAE features that are high precision classifiers for a configuration of squares, such as "white pawn on `e4`, white knight of `f3`". In cases where coverage is higher than reconstruction (such as for `can_check`), it is because there are not many features that are over 95% precision for "there is a *check move* available" from which we can recover if there is an available check move. Coverage is significantly higher because there is at least one feature that has an F_1-score of 0.54 for `can_check`, which may not have a precision greater than 95%.

Table 4: Comparison of performance of linear probes trained to predict board state properties given residual stream activation of ChessGPT after the sixth layer with SAEs evaluated using our coverage and reconstruction metrics.

Concept	**Linear Probe F_1-score**	**Best SAE Reconstruction score**	**Best SAE Coverage score**
`check`	1.00	1.00	1.00
`can_check`	0.93	0.27	0.54
`can_capture_queen`	0.66	0.62	0.48
`queen`	1.00	0.97	0.96
`bishop_pair`	1.00	0.83	0.86
`castling_rights`	1.00	0.98	0.82
`kingside_castling`	1.00	0.98	0.81
`queenside_castling`	1.00	0.97	0.81
`fork`	0.68	0.13	0.38
`pin`	0.67	0.20	0.33
`legal_en_passant`	0.96	0.92	0.90
`ambiguous_moves`	0.72	0.25	0.57
`threatened_squares`	0.96	0.93	0.71
`legal_moves`	0.92	0.66	0.63
`board_state`	0.98	0.67	0.41

Table 5: Comparison of performance of linear probes trained to predict high-level board state properties given residual stream activations with SAEs, both trained on a model with the same architecture as ChessGPT but randomly initialized. Performance on metrics can be high when the metric is correlated with move number or syntax level patterns (such as castling, which corresponds to "`0-0`").

Concept	**Linear Probe F_1-score**	**Best SAE Reconstruction score**	**Best SAE Coverage score**
`check`	0.00	0.00	0.13
`can_check`	0.19	0.03	0.52
`can_capture_queen`	0.00	0.00	0.09
`queen`	0.85	0.95	0.93
`bishop_pair`	0.82	0.74	0.81
`castling_rights`	0.89	0.75	0.65
`kingside_castling`	0.89	0.75	0.65
`queenside_castling`	0.89	0.75	0.64
`fork`	0.01	0.00	0.07
`pin`	0.00	0.00	0.25
`legal_en_passant`	0.00	0.00	0.06
`ambiguous_moves`	0.13	0.00	0.52
`threatened_squares`	0.82	0.73	0.60
`legal_moves`	0.65	0.36	0.45
`board_state`	0.26	0.01	0.11

D. Model Internal Board State Representation

D.1. Othello Models

Previous research of Othello-playing language models found that the model learned a nonlinear model of the board state (Li et al., 2023a). Further investigation found a closely related linear representation of the board when probing for "my color" vs. "opponent's color" rather than white vs. black (Nanda et al., 2023). Based on these findings, when measuring the state of the board in Othello, we represent squares as (`mine`, `yours`) rather than (`white`, `black`).

D.2. Chess Models

Similar to Othello models, prior studies of chess-playing language models found the same property, where linear probes were only successful on the objective of the (`mine`, `yours`) representation and were unsuccessful on the (`white`, `black`) representation (Karvonen, 2024). They measured board state at the location of every period in the Portable Game Notation (PGN) string, which indicates that it is white's turn to move and maintain the (`mine`, `yours`) objective. Some characters in the PGN string contain little board state information as measured by linear probes, and there is not a clear ground truth board state part way through a move (e.g., the "`f`" in "`Nf3`"). We follow these findings and measure the board state at every period in the PGN string.

When measuring chess piece locations, we do not measure pieces on their initial starting location, as this correlates with position in the PGN string. An SAE trained on residual stream activations after the first layer of the chess model (which contains very little board state information as measured by linear probes) obtains a board reconstruction F_1-score of 0.01 in this setting. If we also measure pieces on their initial starting location, the layer 1 SAE's F_1-score increases to 0.52, as the board can be mostly reconstructed in early game positions purely from the token's location in the PGN string. Masking the initial board state and blank squares decreases the F_1-score of the linear probe from 0.99 to 0.98.