Contrastive Sparse Autoencoders for Interpreting Planning of Chess-Playing Agents

Anonymous authors
Paper under double-blind review

Abstract

AI led chess systems to a superhuman level, yet these systems heavily rely on black-box algorithms. This is unsustainable in ensuring transparency to the end-user, particularly when these systems are responsible for sensitive decision-making. Recent interpretability work has shown that the inner representations of Deep Neural Networks (DNNs) were fathomable and contained human-understandable concepts. Yet, these methods are seldom contextualised and are often based on a single hidden state, which makes them unable to interpret multi-step reasoning, e.g. planning. In this respect, we propose contrastive sparse autoencoders (CSAE), a novel framework for studying pairs of game trajectories. Using CSAE, we are able to extract and interpret concepts that are meaningful to the chess-agent plans. We primarily focused on a qualitative analysis of the CSAE features before proposing an automated feature taxonomy. Furthermore, to evaluate the quality of our trained CSAE, we devise sanity checks to wave spurious correlations in our results.

1 Introduction

Chess is one of the very first domains where superhuman AI shined, first with DeepBlue (Campbell et al., 2002) and more recently with Stockfish (Nasu, 2018) and AlphaZero (Silver et al., 2018). While the design of these superhuman programs is intended to gain performances, e.g. by optimising the tree search, the node evaluation or the training procedure, a lot remains to be done to understand the intrinsic processes that led to these performances truly. In this respect, the first component to decipher is thus the DNN heuristic that guides the tree search. While DNNs are often thought of as black-box systems, they learn a basic linear representation of features. During the last decade, arguments to support this hypothesis have been demonstrated repeatedly for language models (Mikolov et al., 2013; Burns et al., 2022; Tigges et al., 2023) but also vision models (Radford et al., 2015; Kim et al., 2017; Trager et al., 2023) and others (Nanda et al., 2023; Rajendran et al., 2024). This strong hypothesis also transferred to chess (McGrath et al., 2022), showing that traditional concepts like "attacks" or "material advantage" were linearly represented in the latent representation of the model.

In this work, we focus on the open-source version of Alpha Zero, Leela Chess Zero (Pascutto, Gian-Carlo and Linscott, Gary, 2019), interpreting the neural network heuristic in combination with the tree search algorithm. In particular, we extend the dynamic concepts introduced in Schut et al. (2023). We state our contributions as follows:

- New dictionary architecture to encourage the discovery of differentiating features between latent representations
- Automated sanity checks to ensure the relevance of our dictionaries
- Discovery and interpretation of new strategic concepts creating a feature taxonomy

Figure 1 summarises our approach and illustrates our aim at disentangling planning concepts. With this paper, we release the code[1] used to create the datasets and to discover and analyse concepts.

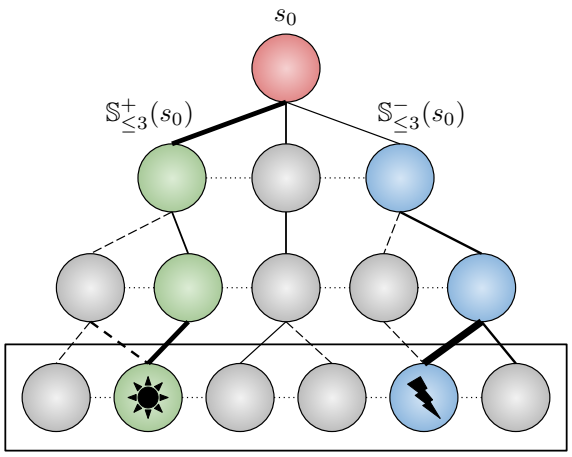


Figure 1: Better viewed in colour. Our proposed framework aims to retrieve planning concepts, represented as icons at the bottom. For that, we analyse the plans of a chess-playing agent. A sampling of an optimal trajectory $\mathbb{S}^-_{\leq 3}(s_0)$ (in green) and a suboptimal trajectory $\mathbb{S}^+_{\leq 3}(s_0)$ (in blue) from a root node s_0. The star represents a concept meaningfully to the optimal trajectory while the lightning represents a concept relevant to the suboptimal trajectory.

2 Background

2.1 Chess Modelling

Heuristic network The studied agent, introduced as AlphaZero (Silver et al., 2018), is a heuristic network used in a Monte-Carlo tree search (MCTS) (Coulom, 2006; Kocsis & Szepesvári, 2006). The network is traditionally trained on self-play to collect data, i.e. the network is frozen and plays against a duplicate version of itself. After the collection phase, the network is trained to predict a policy vector for the next move based on the MCTS statistics and a current state value based on the outcomes of the played games. Here, more specifically, the full network $\mathcal{F}_\theta$, parametrized by θ, can be describe as a tuple,

$$\mathcal{F}_\theta(s) = [\mathcal{P}_\theta(s), \mathcal{W}_\theta(s), \mathcal{M}_\theta(s)], \tag{1}$$

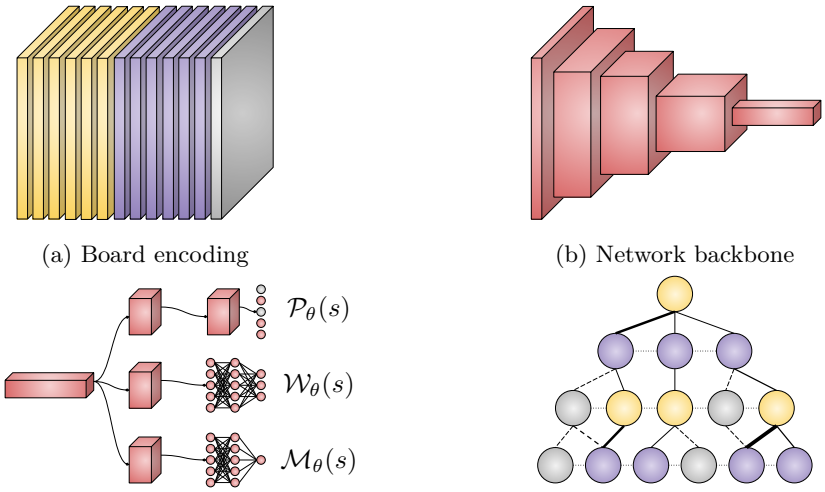
with $\mathcal{P}_\theta(s)$ the policy vector, $\mathcal{W}_\theta(s)$ the win-draw-lose probability and $\mathcal{M}_\theta(s)$ the moves left. The three heads share a Squeeze-and-Excitation (SE) backbone (Hu et al., 2019), based on ResNet (He et al., 2016). The state s fed to the network is made of the current board as well as the 7 previous boards. These boards are decomposed into one-hot planes that we describe in the next paragraphs. The computation process is illustrated in figure 2; for more details, we refer the reader to the exact implementation in (Pascutto, Gian-Carlo and Linscott, Gary, 2019).

Tree-search The AlphaZero (Silver et al., 2018) and its open-source version LeelaZero (Pascutto, Gian-Carlo and Linscott, Gary, 2019) are based on evaluation and tree search similar to Stockfish NNUE. The search algorithm is based on MCTS (Coulom, 2006; Kocsis & Szepesvári, 2006) using a slightly modified version of the upper bound confidence of the PUCT algorithm (Rosin, 2011), equation 2.

$$U(s,a) = Q(s,a) + c_{\text{puct}} \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)} \tag{2}$$

[1] Available in supplementary materials and released upon publication.

Here, we focused on the policy $P(s, a) = \mathcal{P}_\theta(s, a)$ directly outputted by the network. We further detail the computation of the Q-values and their links to the WDL head $W_\theta(s, a)$ and the ML head $M_\theta(s, a)$ in the appendix A.



(a) Board encoding (b) Network backbone

(c) Heads prediction (d) MCTS

Figure 2: Modelling components; first, the boards are encoded into planes (a) and fed to the network backbone (b). The different heads use the extracted features to make heuristic predictions (c) guiding the MCTS when encountering new nodes (d).

2.2 Discovering Concepts

Sparse autoencoders While linear probing (Alain & Bengio, 2018) requires labelled concepts, sparse autoencoders are an efficient tool for discovering concepts at scale without supervision, which were introduced concurrently in Cunningham et al. (2023) and Bricken et al. (2023). The fundamental idea is to decompose the latent activations h on a minimal set of features, formulated as the minimisation of

$$||h - Df||_2^2 + \lambda ||f||_0. \tag{3}$$

D is the feature dictionary and f is the feature decomposition with $f \geq 0$ for the combination view. In practice, sparse autoencoders (SAEs) have been proposed to solve sparse dictionary learning and have already proven to find a wide range of interpretable features (Bricken et al., 2023). In their simplest form, with only one hidden layer, the architecture can be described as

$$f = \mathrm{ReLU}(W_\mathrm{e} h + b_\mathrm{e}), \ \hat{h} = W_\mathrm{d} f + b_\mathrm{d}. \tag{4}$$

Where the encoder weights (W_e, b_e) and decoder weights (W_d, b_d) are trained using an MSE reconstruction loss with l_1 penalisation to incentivize sparsity:

$$\mathcal{L}_\mathrm{SAE} = \mathbb{E}_h \left[||h - \hat{h}||_2^2 + \lambda ||f||_1 \right] \tag{5}$$

We describe in appendix B.2 some additional architectural changes and hyperparameters we used and how we evaluated those.

Dynamical concepts While traditional concepts only rely on a single position (McGrath et al., 2022), dynamical concepts consider sequences of states and are still discoverable using linear probing (Schut et al., 2023). In order to find these concepts, we need to consider an optimal rollout,

according to the chosen sampling method, $\mathbb{S}^{+}_{\leq T}(s_0) = (s^{+}_1, s^{+}_2, ..., s^{+}_T)$ with T being the maximal depth considered starting at state s_0. This rollout is associated with other sub-optimal rollouts $\mathbb{S}^{-}_{\leq T} = (s^{-}_1, s^{-}_2, ..., s^{-}_T)$. A linear probe can then be trained to differentiate the origin set of a state s using the model's hidden state h; the process is illustrated in Figure 1.

3 Methods

3.1 Disantangling Planning Concepts

The basic idea proposed here is to study a latent space vector in contrast with others. The intuition is that we want to know what additional concepts are present in subsequent states. So, for a depth t, we use a pair of vectors defined as a concatenation of the search root s_0 with s^{+}_t from the optimal rollout and s^{-}_t from a suboptimal rollout; similarly to Schut et al. (2023).

$$h^{+} = [h(s_0); h(s^{+}_t)] \tag{6}$$

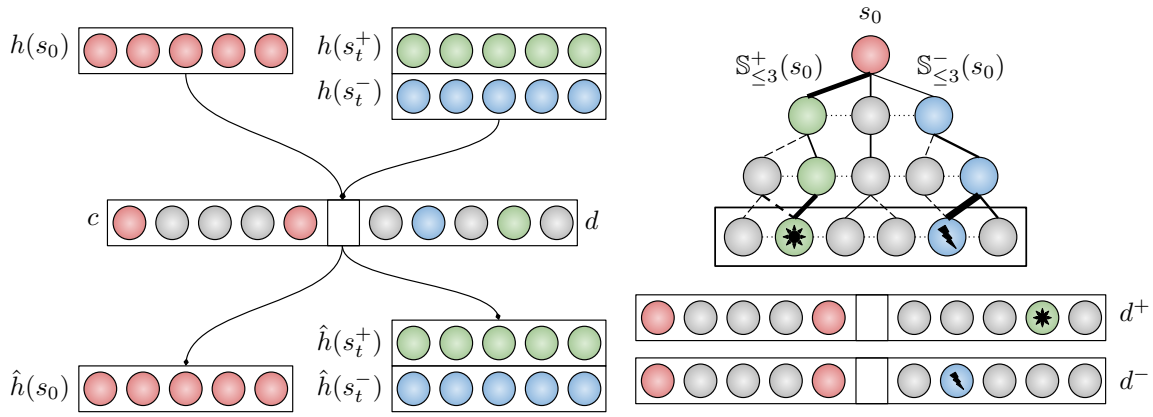
$$h^{-} = [h(s_0); h(s^{-}_t)] \tag{7}$$

We introduce a feature constraint in order to train SAEs with a contrastive loss, equation 8. By dividing the feature dictionary into a set of common features c and a set of differentiating features d, we can separate the s_0 dependence and focus on planning concepts contained in d. In practice, the separation is made using tensor concatenation $f = [c; d]$ as illustrated in the figure 3a.

$$\mathcal{L}_{\text{contrast}} = \mathbb{E}_h \left[||c^{+} - c^{-}||_1 + ||d^{+} \odot d^{-}||_1 \right] \tag{8}$$

In order to concentrate the s_0 dependence into the c-features, we added an additional SAE loss term (reconstruction and sparsity) to reconstruct $h(s_0)$ from c^{+} and c^{-}. Additionally, to ensure that the d-features account for differentiability, we train a linear probe on this intermediate representation of our SAEs using the binary cross-entropy, equation 9. We present the results as part of our first sanity checks in the section 4.1.

$$\mathcal{L}_{\pm} = \mathbb{E}_h \left[-\log \left\{ \mathcal{P}(d^{+}) \right\} - \log \left\{ 1 - \mathcal{P}(d^{-}) \right\} \right] \tag{9}$$



(a) Contrastive SAE

(b) Rollouts concepts extraction

Figure 3: Better viewed in colour. (a) Contrastive SAEs are trained using a contrast of an optimal trajectory (green) and suboptimal trajectories (blue). They take in input the root hidden state $h(s_0)$ and a subsequent node's hidden state $h(s^{\pm}_t)$. The c-features are represented in red, and the d-features are in blue and green. (b) Schematic view of concepts extraction from different rollouts. The dynamical concepts from the rollout $\mathbb{S}^{+}_{\leq 3}(s_0)$ is extracted in d^{+} and for $\mathbb{S}^{-}_{\leq 3}(s_0)$ in d^{-}.

3.2 Concepts Interpretation

Interpreting individual features In order to decipher the nature of the learned dictionary features, a first qualitative analysis can be run using activation maximisation based on data sample (Chen et al., 2020). As illustrated in figure 4, for a given feature, it is possible to investigate the most activated samples. Here, the samples are latent pixels and thus can be visualised on the corresponding chess boards. It is thus possible to create a basic feature categorisation based on the samples they activate in and whether they activate on a wide or restricted range of samples.

Categorising concepts While the learned features appear to be relatively interpretable, it does not scale well with respect to the required human labour. Recent work proposed automated methods to interpret models based on causal analysis (Conmy et al., 2023), using a language model interpreter (Bills et al., 2023) or a multimodal model (Shaham et al., 2024). Yet these methods are hard to supervise humanly and are adding an additional black box layer. We investigate a frugal alternative, creating an automated taxonomy of features using hierarchical clustering. To test this taxonomy, presented in section 4.2, we propose a last sanity check based on the c-features in section 4.1.

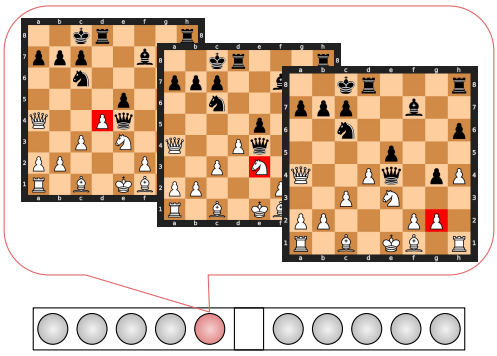


Figure 4: (a) Illustration of the process of interpreting a feature using activation maximisation. The most activated samples are retrieved and analysed. (b) In order to compare a pair of features, the first indicator is the correlation of the feature activation (right). It is also possible to count common samples retrieved using activation maximisation.

4 Experiments

4.1 Sanity Checks

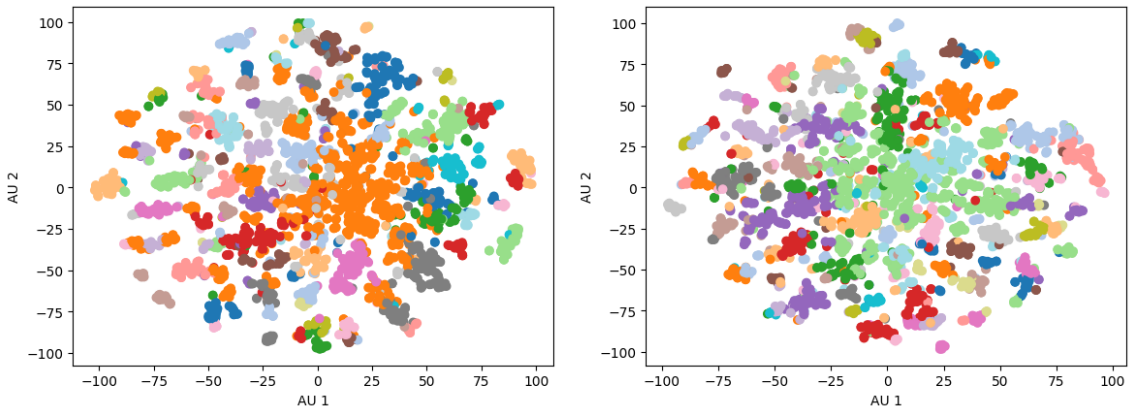
Partitioned features Then, to understand the coarse-grained difference between c-features and d-features, we compute a set of metrics reported in the table 1. The metrics are computed on unseen examples (`test`) similarly to `validation` but were not optimised against. We reported additional metrics in the appendix B.2.

Metric	$F < 0.1\%$	$F > 10\%$	$H(A_s)$	$H(A_t)$	$F_1(\mathcal{P})$	$P(\mathcal{P})$	$R(\mathcal{P})$
c-features	153	58	2.18	2.81	0.537	0.541	0.534
d-features	0	119	2.33	3.24	0.566	0.575	0.557
f	153	177	2.25	3.02	0.578	0.584	0.571

Table 1: Sanity check metrics. F is the feature activation frequency, H is the entropy, and A_s (respectively A_t) is the activation rate on the different squares (respectively trajectories). $\mathcal{P}$ is a linear probe trained to differentiate optimality, with F-score (F_1), precision P and recall (R).

We report more dead (frequency $F < 0.1\%$) c-features, i.e. an over-specification of the c-features, and more overactive (frequency $F > 10\%$) d-features, i.e. over-generalisation of d-features. We see that the entropy $H(A_s)$, the entropy of activation distribution over the square, and respectively $H(A_t)$, the entropy over the trajectories, is smaller for c-features, especially for trajectories. The c-features have overfitted certain trajectories, making them sort of look-up tables. Finally, we train a linear classifier to find the difference between activations originating from optimal or suboptimal trajectories. Notably, the probe $\mathcal{P}$ performances are better using c-features than d-features.

Correlation of features To further compare the c-features and d-features, we clustered the samples using either of them. The visualisation, figure 5, looks alike for both, but the attribution of classes is uncorrelated, with a maximum person coefficient per cluster pair averaging over 0.1.



(a) c-features clustering

(b) d-features clustering

Figure 5: Agglomerative clustering of the `test` samples after an NMF followed by a t-SNE for the visualisation (van der Maaten & Hinton, 2008; Pedregosa et al., 2011). We present the first 100 clusters, and colours are repeated. Each colour represents 5 different clusters, and the colours are independent of (a) and (b). While the structures are similar (due to the t-SNE projection), the labels are uncorrelated, suggesting a difference in representations for the c-features and d-features.

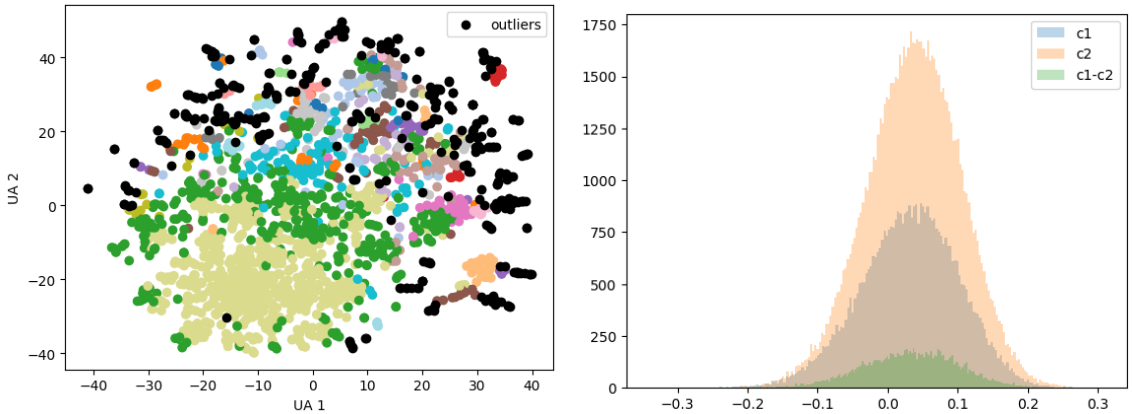
To categorise the two clusterisation approaches, we explored the cluster specificity with respect to the square, state optimality, and trajectory. For that, we computed the respective entropy H_s, H_o, and H_t for each cluster, reported in table 2. We found no clear distinction between the two clusterisations. This informs us that both sets of features contain overspecific features that should be removed, as reported in appendix E, but overall, they can be used in combination.

Metric	H_s	H_o	H_t
c-features	2.2 ± 1.0	2.5 ± 1.3	0.57 ± 0.23
d-features	2.53 ± 0.92	2.9 ± 1.1	0.62 ± 0.17

Table 2: Entropy measures across the clusters of figure 5 (mean and standard deviation).

4.2 Dynamic Concept Clustering

We present a way to explore features by grouping them. For that, we used an agglomerative clustering of features and reported the results in figure 6. It seems here that a lot of features are outliers, but overall clusters appear. We found that the cluster can be found on the activation patterns of the feature, but it is not possible to use the feature vectors, i.e., the columns of W_d. Finally, we report a dendrogram in figure 7, i.e. an automated taxonomy of our elicited features. This analysis could be leveraged to adopt a more or less-grained view of the feature dictionary and thus explore it easily.



(a) Clustered features

(b) W_d cosine similarities

Figure 6: (a) Clustering of the elicited features using an agglomerative clustering approach after an NMF followed by a t-SNE for the visualisation. (b) Cosine similarities of feature vectors originating from two significant clusters. There is no correlation between the intra and extra-cluster similarities.

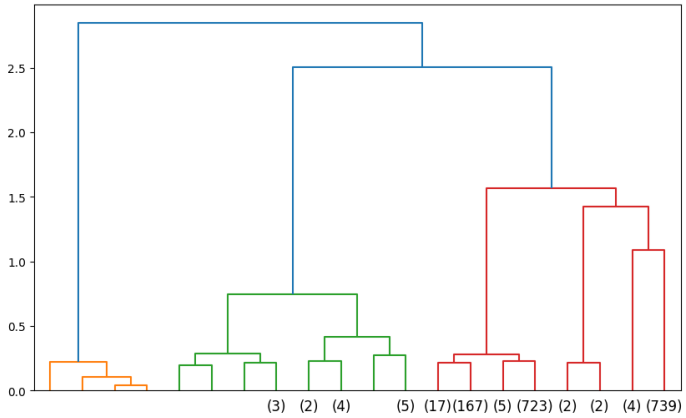


Figure 7: Dendrogram of the clustered features. The dendrogram can help visualise features and be leveraged to explore and interpret groups of features.

5 Discussion

5.1 Limitations

Having good SAEs SAE is still an active field of research, and there is an ongoing effort to find better training strategies and extract the most knowledge from them. It has also proven to be a challenge in this article. We present certain unwanted features in appendix E.

Feature interpretation In order to interpret the features, human analysis cannot be totally replaced. We presented automated analyses in addition to our qualitative results, and we are excited about automated interpretability methods. Yet, having a human in the loop might be the only way not to defer to yet another black box, especially if it requires expert knowledge.

Contrastive interpretations Here, we didn't focus all our attention on finding contrastive interpretations, e.g. comparing the heatmap obtained on the root board and the trajectory board. Yet they might be more prominent, naturally emerging from our contrastive architecture. Thus, we

should aim to interpret the features in a pair of root and trajectory visualisation. In this respect features also show a blinking problem, i.e. features can have a different facet for white and black. Indeed, two similar boards will be encoded quite differently for white and black since the board is flipped for black. Because of this, we might need to pair black root boards with black trajectory boards.

5.2 Future Work

Concept sampling While we presented our sampling results in the appendix B, our choices might have introduced inductive biases. It would be important to quantify the impact of different strategies for suboptimal sampling. For example, it is unclear to what extent the pairing strategy should take deeper trajectory boards and to what extent optimal and suboptimal trajectories can share a common state path.

Weak-to-strong generalisation We already mentioned that using a pair of latent activations is a more flexible interpretability method. But to go further, it is also possible to use the latent activation of smaller models to explain bigger models' strategies, as depicted by Burns et al. (2023). While we only covered an introductory analysis, we think this track is highly promising and relevant to the safety of such models.

Different architectures A direct extension of this work would be to apply the same methodology to a model with the same architecture but a different number of layers. The scaling law could be compared across models w.r.t. the ELO and layer. Furthermore, it would be interesting to use SAEs with a common feature dictionary and a specific encoder and decoder layer for each layer and checkpoint to compare feature transferability.

6 Related Work

Discovering concepts in DNNs Linear probing is a simple idea where you train a linear model (probe) to predict a concept from the internals of the interpreted target model (Alain & Bengio, 2018). The prediction performances are then attributed to the knowledge contained in the target model's latent representation rather than to the simple linear probe. In practice, a lasso formulation, i.e. l_1 penalty, has been a default choice as it encourages sparsity (Tibshirani, 1996), and has been augmented as sparse probing for neuron attribution (Gurnee et al., 2023). Linear probing has also been derived with concept activation vectors (Kim et al., 2018), which often require training a linear probe (Dreyer et al., 2023).

Explaining chess models Chess has always been a good playground for AI, and explanability is no exception (McGrath et al., 2022). Simplified versions of this game have even been created to make research easy (Hammersborg & Strümke, 2023b;a). It is even possible to explore planning, including tree search, through dynamical concepts (Schut et al., 2023).

Explainable tree search It is possible to make tree search explainable by default. By extracting a policy using a surrogate model (Soemers et al., 2022) or using a simpler heuristic model (Soemers et al., 2019).

7 Conclusion

This article explored multiple approaches to gaining knowledge from superhuman chess agents. We designed principles to try to elicit knowledge from the neural network's latent spaces. We successfully found interpretable features that were linked to the model plans. Furthermore, we proposed an automated feature taxonomy to help explore features, keeping a human in the loop. While presenting our key results, we also showed automated sanity checks. Finally, we presented the limitations and possible future directions to tackle them or to continue this project.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *ArXiv*, abs/2212.03827, 2022. URL https://api.semanticscholar.org/CorpusID:254366253.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, Jeff Wu, and OpenAI. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *ArXiv*, abs/2312.09390, 2023. URL https://api.semanticscholar.org/CorpusID:266312608.

Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134 (1):57–83, 2002. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(01)00129-1. URL https://www.sciencedirect.com/science/article/pii/S0004370201001291.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772782, December 2020. ISSN 2522-5839. doi: 10.1038/s42256-020-00265-z. URL http://dx.doi.org/10.1038/s42256-020-00265-z.

Tom Conerly, Adly Templeton, Trenton Bricken, Jonathan Marcus, and Tom Henighan. Update on how we train saes. 2024. URL https://transformer-circuits.pub/2024/april-update/index.html.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *ArXiv*, abs/2304.14997, 2023. URL https://api.semanticscholar.org/CorpusID:258418244.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, 2006. URL https://api.semanticscholar.org/CorpusID:16724115.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *ArXiv*, abs/2309.08600, 2023. URL https://api.semanticscholar.org/CorpusID:261934663.

Maximilian Dreyer, Frederik Pahde, Christopher J. Anders, Wojciech Samek, and Sebastian Lapuschkin. From hope to safety: Unlearning biases of deep models via gradient penalization in latent space, 2023.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=JYs1R9IMJr.

Patrik Hammersborg and Inga Strümke. Information based explanation methods for deep learning agents–with applications on large open-source chess models. *arXiv preprint arXiv:2309.09702*, 2023a.

Patrik Hammersborg and Inga Strümke. Reinforcement learning in an adaptable chess environment for detecting human-understandable concepts. *IFAC-PapersOnLine*, 56(2):9050–9055, 2023b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:51737170.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav), 2018.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

Thomas McGrath, Andrei Kapishnikov, Nenad Tomaš ev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in AlphaZero. *Proceedings of the National Academy of Sciences*, 119(47), nov 2022. doi: 10.1073/pnas.2206625119.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *North American Chapter of the Association for Computational Linguistics*, 2013. URL https://api.semanticscholar.org/CorpusID:7478738.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *ArXiv*, abs/2309.00941, 2023. URL https://api.semanticscholar.org/CorpusID:261530966.

Yu Nasu. Nnue efficiently updatable neural-network based evaluation functions for computer shogi. *Ziosoft Computer Shogi Club*, 2018.

Pascutto, Gian-Carlo and Linscott, Gary. Leela chess zero, 2019. URL http://lczero.org/.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL https://api.semanticscholar.org/CorpusID:11758569.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, J'anos Kram'ar, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. 2024. URL https://api.semanticscholar.org/CorpusID:269362142.

Goutham Rajendran, Simon Buchholz, Bryon Aragam, Bernhard Schölkopf, and Pradeep Ravikumar. Learning interpretable concepts: Unifying causal representation learning and foundation models. *ArXiv*, abs/2402.09236, 2024. URL https://api.semanticscholar.org/CorpusID:267657802.

Christopher D. Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61:203–230, 2011. URL https://api.semanticscholar.org/CorpusID:207081359.

Lisa Schut, Nenad Tomasev, Tom McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero, 2023.

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent. 2024. URL https://api.semanticscholar.org/CorpusID:269293025.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Dennis J. N. J. Soemers, Éric Piette, and Cameron Browne. Biasing mcts with features for general games. *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 450–457, 2019. URL https://api.semanticscholar.org/CorpusID:84842738.

Dennis J. N. J. Soemers, Spyridon Samothrakis, Éric Piette, and Matthew Stephenson. Extracting tactics learned from self-play in general games. *Inf. Sci.*, 624:277–298, 2022. URL https://api.semanticscholar.org/CorpusID:255326863.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *ArXiv*, abs/2310.15154, 2023. URL https://api.semanticscholar.org/CorpusID:264591569.

Matthew Trager, Pramuditha Perera, Luca Zancato, Alessandro Achille, Parminder Bhatia, and Stefan 0 Soatto. Linear spaces of meanings: Compositional structures in vision-language models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15349–15358, 2023. URL https://api.semanticscholar.org/CorpusID:257766294.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL https://api.semanticscholar.org/CorpusID:5855042.

Appendix

A Additional Chess Modelling Details

Board encoding The current position is encoded using planes, formally channels, equivalent to the colours in images, in a tensor of the shape $112 \times 8 \times 8$. The 112 planes can be first decomposed into two parts, the first 104 planes corresponding to the history planes (8 last boards) and 8 additional planes encoding the game metadata. Each board of the history is encoded through 13 distinct planes, comprising two sets of 6 sparse planes each for the current[2] player's and the opponent's pieces, as illustrated in figure 2a. The last 8 planes are always full planes and represent meta information like the castling rights, the current player's colour and the half-move clock value.

[2]Note that the player is the same for all 8 boards of the history.

Move encoding The policy outputted by the network is a vector of size 1858. This number is obtained considering each starting position and counting all accessible ending positions using queen and knight moves. The different promotions should also be accounted for, with promotion to knight being the default in lc0. Note that as the corresponding moves are relative to the swapped board, promotion is only possible at rank 8. This table is hardcoded within the chess engine for programming efficiency and readability.

Tree-search In practice, the Q-values $Q(s, a)$ are obtained through the value $V(s + a)$, and by adding the move-left-head utility $M_\theta(s + a)$ defined in equation 10. The value is simply computed using the network outputted probabilities and the defined reward $\mathcal{W}_\theta(s + a) \cdot R$. These engineering tricks make the network tuning flexible, e.g., to incentivise drawing or aiming for short games.

$$M(s + a) = \text{sign}(-V(s + a)) \cdot \Pi_{m_{\max}} \left[m \cdot (\mathcal{M}_\theta(s + a) - \mathcal{M}_\theta(s)) \right] \cdot \chi \left[\widetilde{V}(s + a) \right] \tag{10}$$

With χ a second-degree polynomial function and $\widetilde{V}$ the extra-value ratio defined as:

$$\widetilde{V}(s + a) = \text{ReLU} \left(\frac{|V(s + a)| - V_{\text{threshold}}}{1 - V_{\text{threshold}}} \right) \tag{11}$$

Here, the final bound used, equation 12, doesn't rely on the visit could $N(s, a)$. It thus can be used with the raw output of the neural network to perform the sampling.

$$U(s, a) = \alpha V(s + a) + \beta M(s + a) + \gamma \mathcal{P}_\theta(s, a) \tag{12}$$

B Technical Details

B.1 Dynamical Concepts Dataset

Chess boards dataset In order to train the SAEs, we created a base dataset[3] of around 20k games from the TCEC archives. These games were then processed and transformed into 20M individual boards to form the board dataset[4]. The first moves were filtered only to take position after the "book exits" and after at least 20 plys. For this preliminary study, we sampled trajectories from 200k random boards for the `train` split and 20k boards in the `test` split. The sampling of trajectories is further detailed below.

Concept sampling In order to choose the best strategy, i.e. the best hyperparameters of equation 12, we run several matches between the different models and hyperparameters; the results are reported in table 3. Using this strategy, we then constructed a trajectory dataset[5] for each model. This dataset was then converted into an activation dataset[6] to make the SAE training easy to configure. When sampling suboptimal trajectories, we used a normalised distribution without any optimal action.

[3]Released upon publication.
[4]Released upon publication.
[5]Released upon publication.
[6]Released upon publication.

		Win rate vs $\mathcal{P}_\theta(s)$				
	Model	1893	3051	4012	4238	Average
Strategy	Raw Q-values: $\mathcal{W}_\theta(s+a) \cdot R$	-0.18	-0.48	-0.73	-0.78	-0.55 ± 0.24
	$U(s,a)$ ($\alpha = 1$, $\beta = 0$, $\gamma = 0.25$)	-0.17	-0.45	-0.65	-0.63	-0.48 ± 0.19
	$U(s,a)$ ($\alpha = 1$, $\beta = 0$, $\gamma = 0.5$)	-0.10	-0.35	-0.67	-0.48	-0.40 ± 0.21
	$U(s,a)$ ($\alpha = 1$, $\beta = 0$, $\gamma = 1$)	0.03	0.03	-0.13	-0.15	-0.05 ± 0.09
	$U(s,a)$ ($\alpha = 1$, $\beta = 0.5$, $\gamma = 0$)	-0.18	-0.57	-0.73	-0.68	-0.54 ± 0.22
	$U(s,a)$ ($\alpha = 1$, $\beta = 0.5$, $\gamma = 0.1$)	-0.20	-0.43	-0.72	-0.68	-0.51 ± 0.21
	$U(s,a)$ ($\alpha = 1$, $\beta = 0.5$, $\gamma = 0.25$)	-0.07	-0.37	-0.67	-0.65	-0.44 ± 0.25
	$U(s,a)$ ($\alpha = 1$, $\beta = 0.5$, $\gamma = 0.5$)	-0.12	-0.33	-0.55	-0.43	-0.36 ± 0.16

Table 3: Hyperparameters tournament scores against the raw policy baseline. Only the combinations selected after an initial random search are reported. Here, the policy is better for almost all models and combinations.

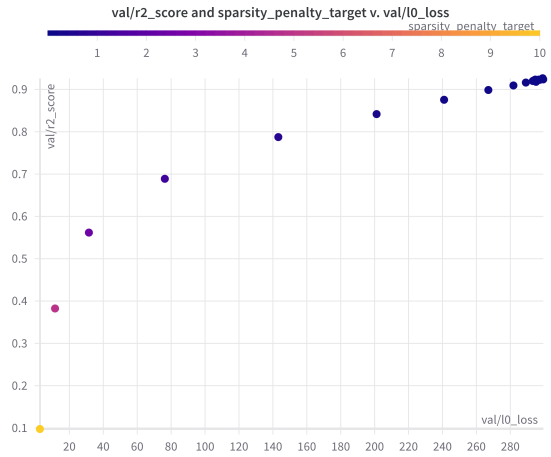
B.2 SAE Training

Procedure We based our SAE training on recent work from like Rajamanoharan et al. (2024) and take into account the monthly updates of Anthropic like Conerly et al. (2024). We will be reporting relevant metrics for our SAEs in the figure 8. $\beta_1 = 0$ stabilised the training. We also use the modified loss, described in equation 13, in order to prevent arbitrary norm of dictionary columns that trick the ℓ_1 norm. Indeed, without it, the features f can get a low ℓ_1 norm but not a low ℓ_0 norm since even small features can reconstruct the activation x if W_d is unconstrained.

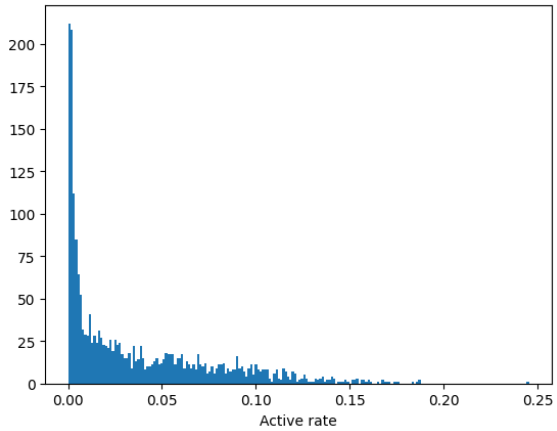
$$\mathcal{L}_{\text{SAE}} = \mathbb{E}_h \left[||h - \hat{h}||_2^2 + \lambda \sum_i |f_i| \cdot ||W_{di}||_2 \right] \tag{13}$$

We will release our trained assets[7]. To make the SAE analysis easy, we also will release the feature activation dataset[8] which will be then used in our interactive demonstration[9]. Hyperparameters are chosen to balance the trade-off between sparsity and reconstruction accuracy, as presented in the figure 8a. We also monitor the activation of the feature, reported in figure 8b, and as already discussed in the section 4.1.

[7]Released upon publication.
[8]Released upon publication.
[9]Released upon publication.



(a) Trade-off sparsity/accuracy



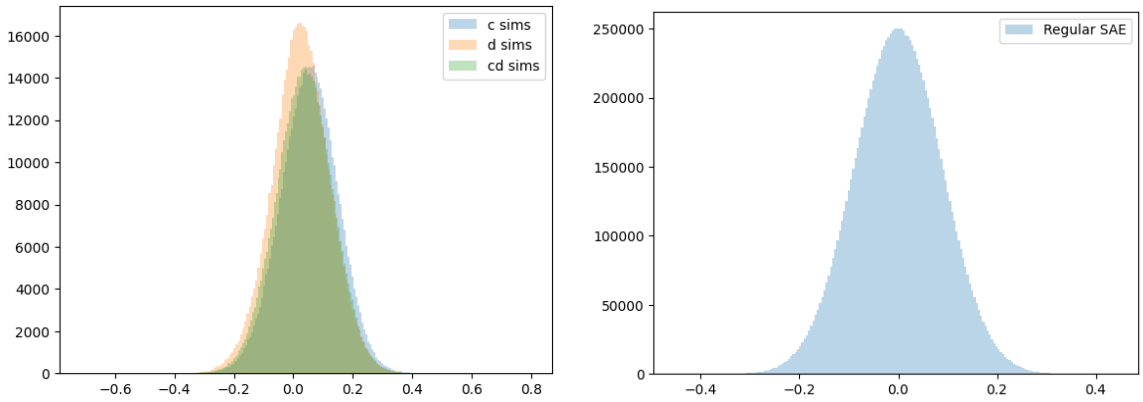
(b) Feature activation histogram

Figure 8: (a) Trade-off between the coefficient R^2 measuring the reconstruction accuracy vs the norm ℓ_0 of the features, measuring the sparsity. The plot is obtained using a sweep of the coefficient λ and shows a power law dependence. (b) The histogram of feature activation rate F. As already pointed out by previous works on SAE, a low-frequency cluster naturally emerges.

Results When training SAEs, the first metrics to report, in addition to the losses, are the ℓ_0 norm of features and the determination coefficient R^2 for the reconstruction. Indeed, we aim to jointly minimise the norm ℓ_0 to get a sparse decomposition and maximise R^2 to ensure a correct reconstruction of the activations. We showed in the table 4 the different metrics obtained for the model used in this article. In particular, the trained SAE has, on average, 73 active features while trying to reconstruct activations of dimension 256, a reduction of around 71%. But with respect to the dictionary, it represents only 3.5% of active features.

Losses	MSE	Sparsity	$\mathcal{L}_{\text{contrast}}$	ℓ_0	R^2
`train`	21.7	26.7	10.7	73.3	0.81
`validation`	21.8	26.8	10.7	73.4	0.81

Table 4: Losses and metrics obtained for the model used in this article for the sets `train` and `validation`. MSE refers to the mean squared error, e.g. the reconstruction loss $\mathbb{E}_h\left[||h-\hat{h}||_2^2\right]$, and similarly Sparsity refers to $||f||_1$. ℓ_0 and R^2 are metrics that were optimised using the `validation` set. ℓ_0 measures the feature sparsity and R^2 the activation reconstruction (1 is the best). As ℓ_0 is a count, it can be understood knowing that the activation dimension is 256 and the dictionary dimension is 2048.



(a) CSAE (b) SAE

Figure 9: Histogram of the cosine similarities of the dictionary vectors. (a) is reported for our CSAE and (b) for a regular SAE. We find that we conserve the independence of the learned directions.

C Concepts in Different Models and Layers

Comparing features by pair It is important to investigate the correlation between features, which is a simple proxy to understand basic interactions between features. This analysis can be run for the c-features and the d-features, which is illustrated in figure 10. We first present a sanity check on the c-features in section 4.1 and expand d-features categorisation in 4.2. This method is especially relevant when dealing with different latent spaces, e.g. from different models or layers. In the following paragraph, we present a small investigation of the correlation between features from different layers and at different training stages.

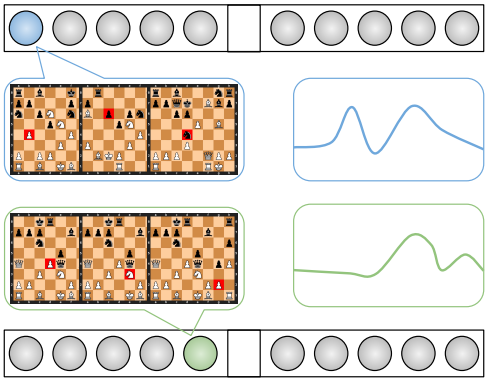
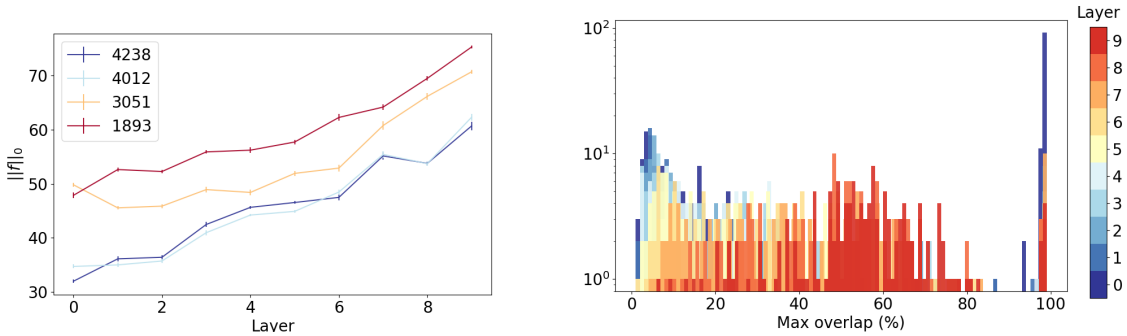


Figure 10: In order to compare a pair of features, the first indicator is the correlation of the feature activation (right). It is also possible to count common samples retrieved using activation maximisation (left).

Probing across different latent spaces In order to investigate universal concepts shared across models or layers we need to probe different latent spaces. A quick analysis of these latent spaces yields that they differ, at least in barycentre, amplitude, and principal components. We thus only investigate the correlation between features and leave the design of universal SAEs decomposing multiple latent spaces simultaneously for future work. Similarly to Bricken et al. (2023), to analyse

features of different SAEs, we used the correlation of the activations to which we add the correlation between the most activated sample, i.e. using data-based activation maximisation Chen et al. (2020).

Feature comparison The study was on a 10-layer model across 4 checkpoints named after their ELO, i-e, their chess performance level; the results are shown in figure 11. While conclusions must be drawn with care, Figure 11(a) seems to show a scaling law of feature density or storage across layers and training. Later latent spaces are denser, surely due to refined and more complex information, but the training compresses the latent spaces, possibly using sharper features. Figure 11(b) represents the correlation between maximum activated samples between the last layer of ELO-4238 and the layers of ELO-4012 and indicates that earlier layers wield more universal features.



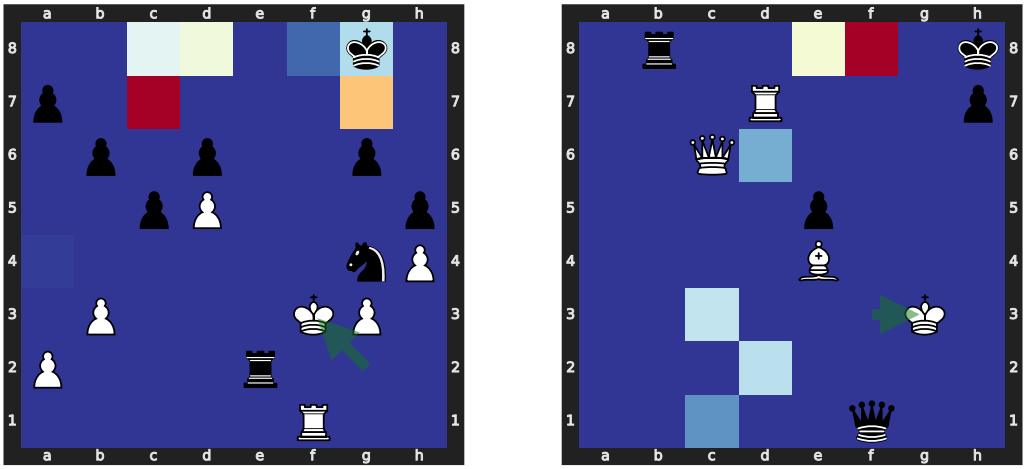
(a) $||f||_0$ across layers for different models named after their ELO.

(b) Overlap for ELO-4012 with the last layer of ELO-4238.

Figure 11: Feature analysis of the agents' latent spaces, summarising scaling properties. The SAEs trained for this figure are regular ones (without the contrastive framing). (a) represents the evolution of ℓ_0 on different models and at different layers. There seems to be a general trend of information densification through layers but more condensed in better models. (b) represent the correlation between features of different layers. While the gradual correlations is expected to correlate with layers, the peak at 100% could indicate over-active features or universal ones.

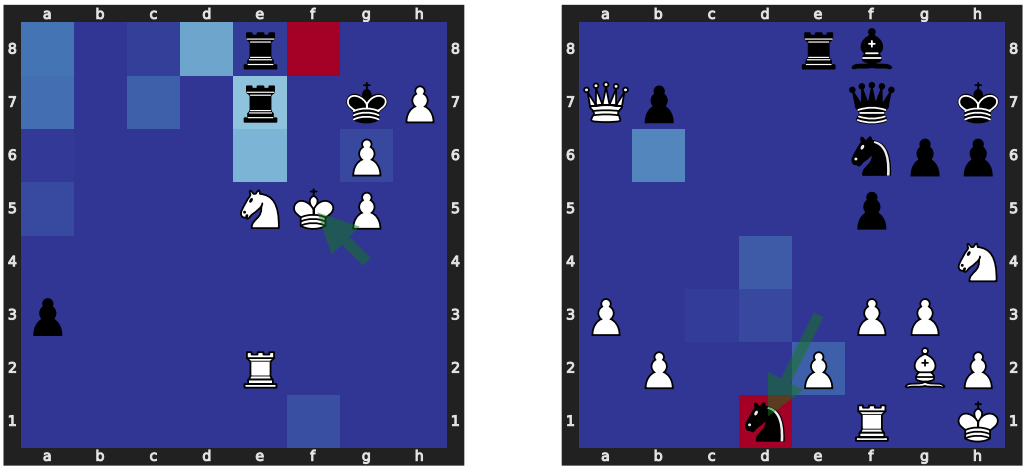
D Feature Interpretation

Qualitative Concept Analysis We cherry-picked features and the samples that maximally activated them to present qualitative analyses. The samples are selected by finding the maximally activating channels and computing the feature on their full board. We first present in the figure 12 a feature that seemed to be linked to the pieces' safety. And we then present a rook threat feature in figure 13.



(a) Safe place (b) Protection

Figure 12: Illustration of a feature linked with the concept of protection. These samples were among the 16 samples that most activated the feature. On (a), the feature is activated on the king and a traditional safe place for the king. The path for the king to join the place is also activated. In (b), the black king is dangerously threatened, and a safe move might be to bring back the queen.



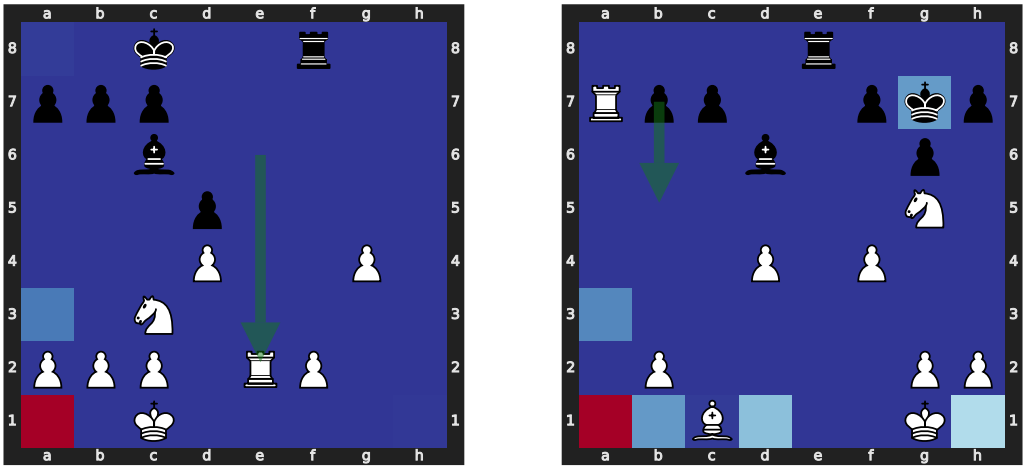
(a) Rook threat 1 (b) Rook threat 2

Figure 13: Illustration of a feature that seems to be linked with the concept of rook threat These samples were among the 16 samples that most activated the feature. The feature activates for both black and white. In (a), the black rook should move to the red square to check the king, while in (b), the white rook should take the knight.

E Unwanted Features

We show two kinds of unwanted features that are present in our trained SAE.

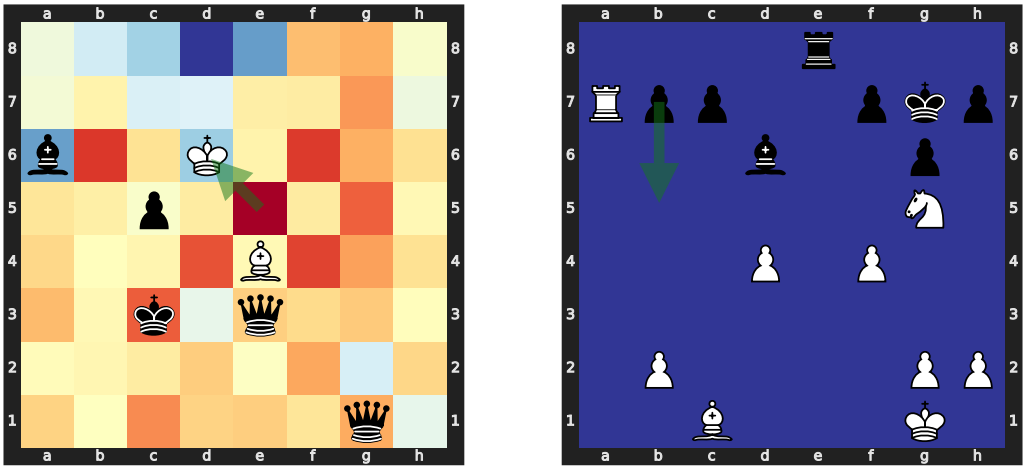
Square specific features Features that are specific to a given square. They act as over-generic features.



(a) White facet (b) Black facet

Figure 14: Illustration of a feature that is linked to the lower left square (a1). (a) was among the 16 samples that most activated the feature, and (b) was chosen arbitrarily. The feature is sometimes dead or differently activated but mostly activates on a1. It also happens to activate on a8 relatively when the heatmap is when the heatmap is flipped according to the model's view.

Trajectory specific features Features that are specific to a given trajectory. They act as lookup tables.



(a) Specific trajectory state (b) Protected square

Figure 15: Illustration of a feature that is linked to a particular trajectory. (a) was among the 16 samples that most activated the feature, and (b) was chosen arbitrarily. On (a), the feature is activated on almost every square, but on (b), it is dead.