[23] Neel Nanda. Actually, othello-gpt has a linear emergent world model, mar 2023. *URL< https://neelnanda.io/mechanistic-interpretability/othello*, 2023.

[24] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[25] Chris Olah, Adly Templeton, Trenton Bricken, and Adam Jermyn. April update. `https://transformer-circuits.pub/2024/april-update/index.html`, 2024. URL `https://transformer-circuits.pub/2024/april-update/index.html`.

[26] Chris Olah, Nicholas Turner, Adam Jermyn, and Joshua Batson. July update. `https://transformer-circuits.pub/2024/july-update/index.html`, 2024. URL `https://transformer-circuits.pub/2024/july-update/index.html`.

[27] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[28] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=UGpGkLzwpP`.

[29] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

[30] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, pages 206–215, 2019.

[31] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. `https://transformer-circuits.pub/2024/scaling-monosemanticity/`, May 2024. Accessed on May 21, 2024.

[32] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

A Appendix

A.1 Glossary of Terms

Sparse Autoencoders (SAEs): Neural networks trained to reconstruct their input while enforcing sparsity in their hidden layer. In the context of this paper, SAEs are used to decompose the dense activations of language models into more interpretable features.

SAE error term: When inserting a SAE into the computation path of the model, errors in SAE reconstruction will propagate to later parts of the model and can change the model output. We refer to the error as the SAE error term, and corresponds to the difference between the SAE output and the original SAE input activation. Marks et al. [21] introduced the idea of adding this error term back to the SAE output to ensure that the SAE does not change model output.

Latent: We refer to neurons in the hidden layer of a SAE as latents to avoid overloading the term "feature". This is in contrast to earlier work which used the term "feature" to refer to both human-interpretable concepts and SAE hidden layer neurons.

Feature: We use the term "feature" to refer to an idealized human-interpretable concept that the model represent in its activations and which a SAE latent may or may not represent.

Monosemantic: Referring to a feature or representation that corresponds to a single, clear semantic concept. In the context of SAEs, a monosemantic feature would ideally capture one interpretable aspect of the input.

Interpretable: A latent being interpretable is not well defined in the field, making it difficult to ensure that different authors mean the same thing when referring to SAE interpretability. When we refer to a SAE latent as being interpretable in this work, we mean that it should behave in line with how it appears to behave after inspecting its activation patterns. If an SAE latent appears to track a feature X by a reasonable inspection of its activations but has subtle deviations from this behavior in reality, we say this is not interpretable. We thus measure interpretability via classification performance when a latent appears to be a classifier over some feature.

Feature dashboard: A dashboard showing activation patterns and max-activating examples for a SAE latent. Feature dashboards are commonly used to interpret the behavior of an SAE latent.

Neuronpedia: A platform, `https://neuronpedia.org`, which hosts feature dashboards for popular SAEs [20].

Token-aligned latent: A latent which seems to roughly fire on variants of the same token. For instance, a "Snake" token-aligned latent may fire on the tokens "Snake", "SNAKE", "_snakes", etc...

Feature splitting: A phenomenon in SAEs introduced by Bricken et al. [2], where a SAE latent tracks a general feature in a narrow SAE, but splits into multiple more specific SAE latents in a wider SAE. For instance, a latent tracking "starts with L" in a narrow SAE may split into a latent tracking "starts with capital L" and a latent tracking "starts with lowercase L" in a wider SAE.

Feature absorption: A problematic form of feature splitting where a SAE latent appears to track an interpretable feature, but that latent has seemingly arbitrary exception cases where it fails to fire. Instead, a more specific latent "absorbs" the feature direction and fires in place of the main latent.

Circuit: In the context of neural network interpretability, a circuit refers to a subgraph of neurons or latents within a neural network that work together to perform a specific function or computation. The study of circuits aims to understand how different components of a neural network interact to process information and produce outputs.

Linear probe: A simple linear classifier (typically logistic regression) trained on the hidden activations of a neural network to predict some property or task. Used to assess what information is linearly decodable from the network's representations.

K-sparse probing: A variant of linear probing where only the k most important latents (as determined by some selection method) are used to train the probe. This helps identify which specific neurons or latents are most relevant for a given task.

Ablation study: An experimental method where a component of a system (in this case, a neuron or latent in a neural network) is removed or altered to observe its effect on the system's performance. This helps determine the causal importance of the component.

Integrated gradients (IG): An attribution method that assigns importance scores to input latents by accumulating gradients along a path from a baseline input to the actual input. In this paper, it's used as an approximation technique for ablation studies.

In-context learning (ICL): A paradigm where a language model is given examples of a task within its input prompt, allowing it to adapt to new tasks without fine-tuning. Often used with few-shot learning techniques.

Residual stream: In the context of transformer architectures, the residual stream refers to the main information flow that bypasses the self-attention and feed-forward layers through residual connections.

Logits: The raw, unnormalized outputs of a neural network's final layer, before any activation function (like softmax) is applied. In language models, logits typically represent the model's scores for each token in the vocabulary.

Activation patching: An interpretability technique where activations at specific locations in a neural network are replaced or modified to observe the effect on the network's output. This helps in understanding the causal role of different parts of the network in producing its final output.

A.2 Proof: absorption decreases SAE loss for hierarchical features

We analyze the effect of a specific form of feature absorption, termed δ-absorption, within a Sparse Autoencoder (SAE) framework. We consider two hierarchically related features, f_1 and f_2 (where $f_2 \subset f_1$), and demonstrate that for a defined family of encoder and decoder weights parameterized by $\delta \in [0, 1]$ ($\delta = 0$ corresponds to no absorption, and $\delta = 1$ corresponds to full absorption):

1. Perfect reconstruction of inputs composed of these features is maintained across all values of δ.
2. The sparsity loss component attributable to these features is a decreasing function of δ, provided the child feature f_2 has a non-zero probability of appearing.
3. Consequently, optimizing for sparsity encourages higher values of δ, i.e., greater absorption.

1. Preliminaries and Assumptions

H1. Dataset and Features Let $\mathcal{D}$ be a dataset. We consider a set of features $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$. Each feature $f_i \in \mathbb{R}^k$ is a vector with unit norm, $\|f_i\|_2 = 1$. Features are mutually orthogonal: $f_i \cdot f_j = \delta_{ij}$ (Kronecker delta), where $\delta_{ij} = 1$ if $i = j$ and 0 if $i \neq j$. An activation $h \in \mathbb{R}^k$ in the model's residual stream is a linear combination of active features: $h = \sum_{j \in \text{ActiveFeatures}} f_j$.

H2. Feature Hierarchy and Probabilities We focus on two features $f_1, f_2 \in \mathcal{F}$ with a hierarchy $f_2 \subset f_1$. This implies that if f_2 is present in a datapoint, f_1 must also be present. The probabilities of observing combinations of f_1 and f_2 are:

- $p(f_1, f_2) = p_{11}$: Probability of f_1 and f_2 co-occurring (e.g., input is $f_1 + f_2$).
- $p(f_1, \neg f_2) = p_{10}$: Probability of f_1 occurring without f_2 (e.g., input is f_1).
- $p(\neg f_1, f_2) = p_{01}$: Probability of f_2 occurring without f_1. By the hierarchy assumption, $p_{01} = 0$.
- $p(\neg f_1, \neg f_2) = p_{00}$: Probability of neither f_1 nor f_2 occurring (e.g., input is 0 or some $f_k, k \neq 1, 2$).

We assume $p_{11} + p_{10} + p_{00} = 1$.

H3. Sparse Autoencoder (SAE) Model The SAE reconstructs an input h as $\hat{h} = f_\phi(h)$. The reconstruction is $\hat{h} = W_d z$, where $z = \text{ReLU}(W_e h)$. No bias terms are used. $W_{e,i}$ is the i-th row of W_e (encoder vector for latent i), and $W_{d,i}$ is the i-th column of W_d (decoder vector for latent i). We analyze two specific latents, z_1 and z_2, intended to capture f_1 and f_2. Other latents z_j for $j > 2$ are assumed to perfectly reconstruct other features f_j (e.g. $W_{e,j} = f_j, W_{d,j} = f_j$) and do not interact with f_1, f_2 due to orthogonality.